Genetic algorithms for optimal reactive power compensation planning on the national grid system

Pilgrim, J. D.

Award date:
2002

Awarding institution:
University of Bath

GENETIC ALGORITHMS FOR OPTIMAL REACTIVE POWER COMPENSATION PLANNING ON THE NATIONAL GRID SYSTEM

Submitted by J. D. Pilgrim, MEng 1st class (Hons)
for the degree of
Doctor of Philosophy
at the University of Bath
2002

Bath, March 21, 2002

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Summary

This work investigates the use of *Genetic Algorithms* (GAs) for optimal *Reactive power Compensation Planning* (RCP) of practical power systems. In particular, RCP of the transmission system of England and Wales as owned and operated by National Grid is considered. The GA is used to simultaneously solve both the siting problem – optimisation of the installation of new devices – and the operational problem – optimisation of preventive transformer taps and the controller characteristics of dynamic compensation devices.

A computer package called *Genetic Compensation Placement* (GCP) has been developed which uses an *Integer coded GA* (IGA) to solve the RCP problem. The RCP problem is implemented as a multi-objective optimisation: in the interests of security, the number of system and operational constraint violations and the deviation of the busbar voltages from the ideal are all minimised for the base (intact) case and the contingent cases. In the interests of cost reduction, the reactive power cost is minimised for the base case. The reactive power cost encompasses the costs incurred from the installation of reactive power sources and the utilisation of new and existing dynamic reactive power compensation devices.

GCP is compared to SCORPION (a planning program currently being used by National Grid) which uses a combination of linear programming and heuristic back-tracking. Results are presented for a practical test system developed with the cooperation of National Grid, and it is found that GCP produces solutions that are cheaper than solutions found by SCORPION and perform extremely well: an improvement in voltage profiles, a decrease in complex power mismatches, and a reduction in M*Volt Amps-reactive* (VAr) utilisation were observed.

Acknowledgements

I would like to thank my supervisor Furong Li for giving me this opertunity, and for her advice and support during this project.

I would also like to thank two of the most fantastic people I know: my Mum and Dad, Jenny and Derek. Their unwavering enthusiasm, support, assistance and friendship will always be an inspiration to me.

Thank you Becca, your love and support has been my strength. James, Matt! Gonna miss living with you two, thanks for a great time. Not forgetting the rest of my world: Barry, Allan, Simon, Steve, Gary, Claire, Tony and Neil. Thanks to you all, and good luck with the rest of your lives, I hope I feature in them somewhere.

Many thanks to Gary, Dan and Andy, for teaching me almost everything I know about linux and power systems.

Thanks to EPSRC and Natioal Grid for supporting this project, and, in particular, thanks to Chris, Ahmad and Hai-Bin for their advice and guidance, which has kept this project on course.

Thanks to Emma, Paul, the computer support team and the rest of the staff at Bath, who've helped in so many ways, on so many occasions.

And thanks to Matt for this quote: "Oh my God: he's using a genetic algorithm!" (VIRTUOSITY 1995 Director: Brett Leonard Writer: Eric Bernt)

Contents

List of Figures

List of Tables

List of Abbreviations

Organisations / Groups

IEEE	Institute of Electrical and Electronics Engineers, ICC.
UK	United Kingdom
CEGB	Central Electric Generating Board
PSMAC	Power Systems Modelling And Control
EPSRC	Engineering and Physical Sciences Research

Units and measurements and physical qualities

AC	Alternating Current
DC	Direct Current
VAr	Volt Amperes, reactive
MVAr	Mega Volt Amperes, reactive
MW	Megawatts
kV	Kilovolts
HV	High Voltage
LV	Low Voltage

Compensation Devices

RP	Reactive Power
FACTS	Flexible AC Transmission System
SVC	Static VAr Compensator
SVS	Solid-State Synchronous Voltage Source
VSC	Voltage Sourced Converter
VSI	Voltage Sourced Inverter

GTO	Gate Turn Off
STATCON	Static Condenser
STATCOM	Static Synchronous Compensator
SSSC	Solid-State Series Compensator
UPFC	Unified power flow converter
TCR	Thyristor Controlled Reactor
TSC	Thyristor Switched Capacitor
TSSC / TSSR	Thyristor Switched Series Capacitor / Reactor
TCSC / TCSR	Thyristor Controlled Series Capacitor / Reactor
TCBR	Thyristor Controlled Braking Resistor
TCPST	Thyristor Controlled Phase Shifting Transformer
LCC	Line Commutated Converter Compensator
TCT	High Impedance Thyristor Controlled Transformer
SR	Saturable Reactor

Software Packages

OPF	Optimal Power Flow
OPFL02	Optimal Power Flow 02
CPF	Complexe Power Flow
SCORPION	Security Constrained Optimal Reactive Power Planning
GAE	Genetic Algorithm Engine
LFI	Load Flow Interface
GCP	Genetic Compensation Placement

Optimisation

RCP	Reactive power Compensation Planning
LP	Linear Programming
SA	Simulated Annealing
GA	Genetic Algorithm
SGA	Simple Genetic Algorithm

IGA	Integer coded Genetic Algorithm
PGA	Protein coded Genetic Algorithm
SUS	Stochastic Universal Sampling
BLX	Blended linear crossover
SBX	Simulated Binary Crossover
MOF	Measure Of Failure

Design

UML	Unified Modelling Language
OOP	Object oriented programming
ABC	Abstract Base Class
XML	eXtensible Markup Language
HTML	Hyper-Text Markup Language
DSP	Digital Signal Processing

Biochemical

PCR	Polymerase Chain Reaction
DNA	Deoxyribonucleic Acid
RNA	Ribonucleic Acid

List of Symbols

L	Maximum inductive output of compensation device (MVAr)
C	Maximum capactive output of compensation device (MVAr)
V	Voltage set point of voltage-ouput controller characteristic (pu)
G	Target voltage of HV side of a preventive tap (pu)
T	Transformer tap setting
$Vdev$	Votlage deviation (pu)
$Pmis$	Real power mismatch (MW)
$Qmis$	Reactive power mismatch (MVAr)
Υ	Number of busbar constraint violations
b	The reference number of a busbar
l	The reference number of a line
B	The number of busbars of a particular system
L	The number of lines of a particular system
$\mathcal{F}$	Fitness value of string
$\check{\mathcal{F}}$	Intermediate fitness value of a string
θ	An objective function
Θ	Set of objective functions
ω	Weighing value
$\mathcal{N}$	the set of natural numbers including zero.

Chapter One

Introduction

1.1 Motivation

In this modern electrical age, an unreliable supply of electrical power, or one of poor quality, would be unacceptable. Every day we rely on the power system operators to maintain the security of the power system and the quality of the power we receive, but this day to day moderation is insufficient in the long term. The changing profile of power generation, the increasing demand from the consumer, and the pressure caused by environmental concerns to maximise the use of existing transmission lines are causing significant, long term, changes in the electrical power flows and voltage profiles of the transmission system. Due to this constantly evolving use of the transmission system, particularly in the UK, steps must be taken to account for these changes – as far as they can be predicted – and, within the scope of these predictions, to implement minimal physical changes to the power system to better facilitate the future transport of electrical power to the consumer. This task falls in the domain of the power system planner [Horwill *et al*, 1991].

The reactive power compensation plays an important role in the planning of power systems [Delfanti *et al*, 2000]. Reactive power is associated with the charging and magnetising currents, which are generated by the capacitive and inductive components of the power system. Virtually all plant on a power system either generates or absorbs reactive power and a close relationship can be shown between reactive power and system voltages: a deficiency of reactive power at a busbar causes the voltage at that point to drop; an excess, and the voltage will rise. Reactive power therefore presents a method for controlling

voltage. By maintaining system voltages within acceptable limits, the quality and reliability of a power system can be maintained [Weedy, 1987].

The reactive power compensation problem faced by power system planners is to ensure that, in the future, there is sufficient reactive power compensation to support the voltage profile of the transmission system, not only during normal operation, but also when the system is suffering the effects of a contingency: for example, the loss of a transmission line.

The problem is inherently one of optimisation: the maximum benefit, and satisfaction of security and quality constraints, must be achieved at the minimum cost. Due to the size and complexity of modern interconnected power systems – and the associated costs of compensation – this is one of the most important and challenging optimisation problems facing those accountable [Lee and Yang, 1998]. The reactive power compensation planning problem has, therefore, received much attention from the optimisation community and many contemporary techniques have been applied to its solution.

The Genetic Algorithm is one such optimisation technique that has been receiving increasing attention, and its adeptness at solving problems of this type has been clearly demonstrated for the siting of non-dynamic compensation [Lee and Yang, 1998; Abdullah *et al*, 1997; Chung and Leung, 1999; Kalyuzhny *et al*, 2000; Delfanti *et al*, 2000; Mantovani *et al*, 2001]. The algorithm draws its inspiration from the fields of natural genetics and Darwinian evolution. It uses a population of solutions on which operators that model the effects of reproduction, mutation, and survival of the fittest, act. These operators explore the problem space and exploit productive regions of it, hence finding optimal solutions to the problem.

This work aims to further investigate the use of the Genetic Algorithm by investigating its ability to optimise problems of a practical scale, in terms of system size and number of contingencies: in particular, the England and Wales power system is studied. Also, siting is carried out for not only

simple fixed output devices – such as fixed capacitors – but also dynamic devices, such as the Static VAr Compensator and the synchronous generator. The algorithm also offers some support for the more contemporary third generation compensation devices such as the STATCOM.

A software package has been designed and implemented which solves the placement problem and results are presented for implementations employing a Simple Genetic Algorithm, an Integer Coded Genetic Algorithm, and extensions thereof. Results show that the GA technique developed herein finds solutions that outperform a linear programming combined with heuristic back tracking technique, which is currently being used for industrial applications by National Grid. Solutions presented are shown to be not only cheaper, but also to reduce voltage deviation, reduce complex power mismatches and reduce MVAr utilisation for all system states considered and hence have a positive impact on quality and security.

1.2 Thesis overview

Chapter 2 introduces the salient aspects of power systems and their operation. In particular, reactive power and reactive power compensation are discussed. The chapter then presents a synopsis of the reactive power compensation devices, arranged in terms of their date of invention and the technologies they exploit. This chapter also introduces tap changing transformers which play an important role in the reactive power compensation problem.

The reactive power compensation placement problem is introduced in chapter 3, and a discussion of the various solutions to this problem that can be found in the literature are presented. Finally, SCORPION, a reactive power planning program currently used by National Grid, is discussed.

Chapter 4 provides a detailed discussion of the optimisation technique inves-

tigated in this work: the *Genetic Algorithm* (GA). The basic mechanisms of the algorithm are detailed, as well as some of the extensions that exist, and the purposes they serve. The concepts behind solution coding and problem expression are presented, as is a discussion of the parameters that can affect the performance of the Genetic Algorithm.

The full problem, as investigated in this work, is specified in chapter 5, including the representation of candidate solutions within the GA and functions for their evaluation.

In the course of this work, a software package has been developed to perform optimal reactive power compensation planning using a GA. This package has been named *Genetic Compensation Placement* (GCP). The design and implementation of GCP is discussed in chapter 6. This chapter also discusses how an object oriented design paradigm was used in the development of GCP, and the benefits this brought.

In addition to the use of a Simple Genetic Algorithm, a number of extensions have been implemented; these are described in chapter 7. Firstly, an *integer coded GA* (IGA) was used and, secondly, an extension of the IGA that draws inspiration from the study of the role and function of proteins.

Results of experiments conducted – using a practical sized problem developed with the cooperation of National Grid – are presented and discussed in chapter 8. The performance of the three GA based approaches are compared, firstly, to each other; a variety of different mutation rates, crossover rates and reproduction rates are used. From these experiments, an optimal GA implementation is defined which uses an integer coding approach. Solutions found by this optimised GA are compared to those of a implemented solution that uses a linear programming technique and heuristic back-tracking: SCORPION, the software used by National Grid. Both implementations attempt to find a solution to the RCP problem that maintains security by ensuring that all system and operational constraints are satisfied whilst minimising the reactive power

cost. The reactive power cost encompasses the costs incurred from the installation of reactive power compensation and utilisation of dynamic reactive power devices.

It was found that this formulation of the objective function tended to produce systems that were highly stressed, although no limits were violated. In particular, many system voltages tended to be approaching their limits. To account for this, a further objective was added to the optimisation: voltage profile improvement. Results are presented for this improved implementation of GCP.

Chapter 9 discusses the conclusions that can be drawn from this work. In particular, the results indicate that GAs perform well when applied to a full, practical sized, multi-contingent RCP problem and can find solutions that outperform SCORPION, a "tried and trusted" solution.

A number of suggestions for future investigations based upon this work and extensions of the solution presented herein are presented in chapter 10.

1.3 Contributions from this research

- Firstly, the practical RCP problem has been fully specified – in terms of the planning and operational sub-problems – which can serve as a reference for future work.
- A solution to the RCP problem has been implemented in the form of a software package, written in C++, which uses a GA to solve the full RCP problem. The software can independently or simultaneously solve both the planning and operational reactive power compensation problems. The software optimises the base case and contingent cases simultaneously rather than sequentially.
- Both fixed and dynamic compensation devices are available for place-

ment in the planning sub-problem and, where applicable, optimisation in the operational sub-problem. Due to the fact that devices may be of arbitrary, discrete sizes, the algorithm can be constrained so that commercialy sized devices only are avialiable for siting.

- The concept of objective switching has been introduced. This technique uses heuristic rules to control which objectives of a multi-objective optimisation problem are used in solution comparison. This enables objectives to be used for the sole purpose of guiding the search into feasible regions. Once feasible solutions are found, these extra objectives are ignored. This means that feasible solutions can then be freed of the evolutionary-intensive act of pursuing multi-optimum solutions. This has been found to work extremely well when considering problem-spaces with large areas of infeasibility, such as contingent power systems.

- A new GA has been developed, the *protein coded GA*, inspired by studies of the role and function of proteins. At the heart of the GA is an integer coded GA using Simulated Binary Crossover [Deb and Beyer, 2001] and a two level mutation operator. The GA uses problem specific knowledge – in the form of *parameter templates* to prevent illegal parameter values from appearing in the strings due to initialisation, crossover and mutation. This is intended to alleviate undesired effects caused by problem space distortion. The parameter templates are also used to form a *meta-string*: a representation of the strings in which floating-point distances along a string can be calculated in terms of the entropy of the representation.

- A multi-objective formulation of the RCP problem has been developed in which candidate solutions seek to optimise the voltage profile, reduce the reactive power cost and prevent any system or operational constraints. By using this multi-objective formulation in conjunction with Pareto optimisation, GCP has been shown to reliably find optimal solutions that provide well-conditioned power systems while still incurring

a low cost.

- The GA planning program developed has been shown to outperform SCORPION, a RCP program used by National Grid, when siting compensation on stressed systems of a practical size under multiple contingencies. SCORPION uses a linear programming technique to solve each system state individually, combined with heuristic back-tracking to reduce net investment across all system states considered. Solutions found by the GA based technique, developed herein, are shown to be not only cheaper than the solution found by SCORPION, but also to reduce voltage deviation, reduce complex power mismatches, and to reduce MVAr utilisation across all system states. This work has shown that GAs are indeed a powerful optimisation technique that can find optimal solutions to practical RCP problems.

Chapter Two

Power systems and reactive power compensation

This chapter starts by presenting an introduction to power systems. Following on from this, section 2.2 provides an in-depth discussion of an aspect of power system operation that is particularly relevant to this project: *reactive power*. Virtually all plant on a power system either generates or absorbs reactive power and a close relationship can be shown between reactive power and system voltages. Reactive power therefore presents a method for controlling voltage. By maintaining system voltages within acceptable limits, the quality and reliability of a power system can be maintained.

This opportunity for controlling voltage through reactive power has been implemented in the form of *reactive power compensation* devices. Section 2.3 introduces the various devices, their operational characteristics and the technologies they exploit.

Due to the fact that the reactive power compensation problem is fundamentally tied in with the problem of voltage control, another class of voltage control devices must also be considered: variable transformer taps. These devices are introduced in section 2.4.

Finally, section 2.5 introduces the crux of the problem addressed by this work: the placement of reactive power compensation devices. Due to the changing requirement for reactive power compensation, combined with the extremely high capital cost of devices and the scale of modern interconnected power systems makes the problem of reactive power compensation planning one of the most interesting and challenging optimisation problems in the field of electrical power systems [Lee and Yang, 1998].

2.1 Introduction to power systems

Electricity is a form of energy that provides a convenient means of converting and conveying energy. At generating stations, kinetic energy – of expanding steam, for example – is transferred to a rotating shaft which drives a generator, consisting of an assembly of electromagnets and conductors, to produce electrical energy by electromagnetic induction. The electrical power can then be transmitted, over transmission networks, to the customer[Chard, 1976; Weedy, 1987].

National Grid owns, maintains and operates the transmission system of England and Wales. The system consists of a main 400 *kilo volt* (kV) network and subsidiary 275 kV networks around major demand centres. It is connected, via Supergrid transformers, to the distribution networks (operated at voltages of 132 kV and below) which are owned by the regional electricity companies. The transmission system is also connected to generators – owned by separate generating companies – and to the French and Scottish systems [Thomas *et al*, 1995]. To give some idea of scale, the National Grid Seven Year Statement [National Grid, 2001] details 602 circuits, and 255 substations. The record peak demand recorded by National Grid was 51,012 *Mega Watts* (MW) recorded on Tuesday 16th January 2001.

Within the electricity supply industry of England and Wales, the regulator separately licenses the roles of generation, transmission and distribution. The transmission businesses are under obligation to operate a secure and economic network as governed by such standards as voltage performance and power quality at distribution system interfaces. The provision of reactive power to the system operator for voltage support is vital in ensuring a secure and stable transmission system, as is discussed in section 2.2. Generators are obligated to maintain a reactive capability within a specified power factor range. It is of note that from the first of April 1998, England and Wales have been pioneering

the world's first reactive power market [Chebbo *et al*, 1999].

2.2 Introduction to reactive power

An *Alternating Current* (AC) electrical transmission system comprises overhead lines, underground cables, transformers and shunt devices (capacitors and reactors). Each of these circuit components has three fundamental electrical characteristics:

- *resistance* quantifies the rate at which electrical power is absorbed by the circuit component and converted into heat;
- *capacitance* quantifies the ability of the circuit component to store electrical charge;
- *inductance* quantifies the ability of the circuit component to create a magnetic field as a result of the electrical current passing through it.

Reactive power is associated with the charging and magnetising currents which are generated by the capacitive and inductive components of the power system. In pure reactive loads, the current waveform is 90 ° out of phase with the voltage across its terminals. In complex loads – those with non-zero reactance and resistance – the current can be resolved into two components: one in phase with the voltage and the other 90 ° out of phase. Reactive power is defined as the product of the out-of-phase current and and instantaneous voltage [Luckett, 1999].

Reactive power is less tangible than real (active) power. Real power represents the transfer of physical energy across an electrical power system and is measured in *Mega Watts* (MW). Whereas real power can be converted into mechanical and other forms of power, reactive cannot. In addition, reactive power is not dissipated in the circuit but flows backwards and forwards across

the system, every half cycle, moving between the electric fields of capacitors and the magnetic field of inductors.

The unit of reactive power is the VAr. Of the conventions for the interpretation of the sign of VArs the following will be adopted in this work:
Capacitive loads generate positive – or lagging – VArs and absorb negative – or leading – VArs, whereas inductive loads generate negative VArs and absorb positive VArs. This is the convention recommended by the International Electrotechnical Commission, and is generally accepted as standard.

The relationship between the scalar voltage difference between two nodes in a network and the flow of reactive power can – approximately – be expressed by:

$$\Delta V = \frac{RP + XQ}{V} \tag{2.1}$$

where

ΔV = voltage difference between two busbars, A and B

P = real power flow from busbar A to busbar B

Q = complex power from busbar A to busbar B

R = line resistance

X = line reactance

V = voltage of busbar A

with a transmission angle, δV, proportional to:

$$\frac{XP - RQ}{V} \tag{2.2}$$

As for most power systems $X >> R$ it can be seen from equation 2.1 that ΔV determines Q, that is, the flow of reactive power along a circuit is determined – and indeed can be controlled by – the difference in voltages of the two ends

of the circuit. Conversely, the voltage difference can be affected by controlling the reactive power flow.

This has important implications for the operation of a power system. If there is a deficiency or surplus of reactive power at a node this deficiency must be supplied from the connecting lines, and hence the voltage at that point will fall or rise, respectively. As the quality and reliability of a power system is ensured by maintaining the load bus voltages within their permissible limits, guaranteeing there is sufficient reactive power available is of utmost importance to a power system operator [Weedy, 1987]. The process of using reactive power to control voltage is often referred to as the use of reactive power to *support* the system voltages. The idea of installing plant on a power system for the purpose of absorbing or generating reactive power is referred to as reactive power *compensation*. The challenging part of ensuring a power system has sufficient compensation is in the changing nature of a system's requirement, as will be discussed in the following section.

2.2.1 The changing requirement for compensation

The reason why the subject of reactive power compensation is still such an active forum is that the profile of power generation and reactive power support are continuously changing, and thus the instantaneous VAr requirement of a system is constantly changing on a number of levels.

- Temporal cycles

  The daily and seasonal changes in the system loading and load type cause changing VAr requirements. For example, during the daytime when power systems are heavily loaded the system voltage decreases. This can be attributed to system plant such as overhead lines, which absorb reactive power when fully loaded and generate reactive power when lightly loaded.

Such changes are, to a fairly high degree, predictable and cover a short time frame. This makes the problem one of *operational* considerations. The reative power dispatch problem itself has also recived attention from the optimisation community, using many of the optimisation techniques described in chapter 3 [Mamandur and Chenoweth, 1981; Ma and Lai, 1995, 1997; Chebbo and Irving, 1995; Chebbo *et al*, 1995]. There is currently no means for installing extra, or moving existing, compensation over the course of a day or even month, so the capability must already be in place to cover the full range of the period's requirement. Having stated that, the current trend is moving towards investing in *relocatable* compensation, that can be physically moved to alternative locations within a power system over time periods of a few months [Luckett, 1999].

- Topographic / profile changes

  Looking at the transmission system within a time frame of years, there are two major categories of changes that must be considered. Firstly, the actual physical structure of the network changes. Generators may be commissioned or decommissioned and the major demand centres change as domestic and industrial areas expand or contract. At the present time, much of the active power (mega watt) generation is in the north of the country, whilst much of the demand is in the south, resulting in a predominant north to south transfer [Horwill *et al*, 1991]. This is, in turn, responsible for the heavy loading of north to south circuits. Secondly, in addition to physical changes, with the privatisation of the CEGB and movement towards free markets for real and reactive power supply, market forces affect the choices of preferred generators, thus shifting the pattern of power flows across the network. These long term changes can be placed in the category of *planning* considerations. The transmission system operator needs to be constantly evaluating and predicting the scope of the future requirement for reactive power, and ensuring that those requirements are met when the time comes.

- Contingencies

  The final category concerns *contingencies*. Within the scope of power systems this generally refers to failure of power system plant. This can range from the loss of a power line – a *line-outage* – to the loss of a generator. The operator needs to ensure that the security and stability of the transmission system are not compromised by such events. Following a contingency the system profile – busbar voltages and line complex power flows – can instantly and significantly change as the loading of lines changes to cover demand. This causes different reactive power requirements to support the voltage.

A further two points affect the requirement for reactive support on the National Grid system. Firstly, the National Grid has little control over the short and long term generation pattern, except through its charges (to generators and consumers) for the use of the transmission system. Secondly, for environmental reasons, it has become desirable to maximise the use of existing transmission wayleaves by improving conductor design to increase thermal ratings. With larger power flows on circuits, the reactive losses on the transmission system are likely to increase. There must be sufficient capability on the system, from generators and compensation plant, to ensure that an acceptable voltage profile can be maintained for normal and contingent states of operation [MacQueen *et al*, 1998].

2.3 Reactive power compensation devices

Virtually all plant on a power system produces or absorbs VArs, but this is generally not useful for the purposes of compensation as the operator has no direct control over it. What is required is the ability to control VArs at particular nodes as needed. Henderson *et al* [1958] investigated the feasibility of supplying system VAr requirements solely from generating stations operating

at off-nominal power factors. By varying the phase difference between the generator voltage and the system voltage, reactive power can be exchanged with the system.

They concluded by stating that the economic viability of transmitting VArs is severely degraded by increases in distance and KW loading, and that over even moderate distances, VAr support at a receiving-end bus is normally required.

There are a number of devices available for the purpose of VAr compensation, each with distinct VAr generation characteristics, and, therefore, functions that they serve. These can be grouped into three categories of devices according to the technologies they represent.

2.3.1 First generation compensation devices

Table 2.1 lists the *first generation* of devices; these are the earliest and simplest compensation devices. Fixed shunt devices are used to correct any basic system requirements; capacitors in particular are connected across the system to provide constant voltage support at busbars. They suffer the drawback of having the reactive power output proportional to the voltage, so that as the voltage drops – when needed most – the output of VArs (be it positive or negative) falls. In addition, on light loads the high voltage causes increased VAr output from capacitors which tends to increase the voltage.

Series capacitors are installed on lines to counteract their natural inductance, but suffer from the high overvoltage produced if they short-circuit.

The switched shunt devices may have one or more reactive elements which can be mechanically switched in and out, providing varied, albeit discontinuous, amounts of compensation. Although more flexible, they cannot adjust the voltage faster than the operating speed of the switch.

Table 2.1: First generation compensation devices

Device name	Abbreviation
Fixed shunt reactor	FR
Fixed shunt capacitor	FC
Mechanically switched shunt reactor	MSR
Mechanically switched shunt capacitor	MSC
Saturated reactor	SR
Series capacitor	SC
Quadrature booster Phase shifting transformer	QB PST
Synchronous compensator	condenser

These devices are used for steady state voltage and reactive power flow control, which helps minimise excessive losses and maintain the voltage profile of the transmission system. Series capacitors also have effects on the stability of the system, helping to maintain the steady state power transfer capacity and voltage stability.

The synchronous condenser introduces dynamic reactive power output, and so can also combat transient and dynamic effects such as voltage variations due to daily load cycles, repetitive impact loads – such as arc furnaces – and sudden changes in loading. They also improve the transient stability of the system which prevents such things as oscillatory dynamic instability [Hauth *et al*, 1982]. A synchronous compensator or condenser is a synchronous motor without a mechanical load, which can be made to generate or absorb VArs by varying its field excitation. The excitation is generally controlled by reference to the system voltage, providing a device that has a voltage-VAr characteristic. Qualitatively, this characteristic is the same for all dynamic compensation devices and is illustrated in figure 2.1. This type of control characteristic causes

such devices to inject capacitive VArs when the voltage is low, which will act to increase the voltage, and inject inductive VArs when the voltage is high, which will act to decrease the voltage.

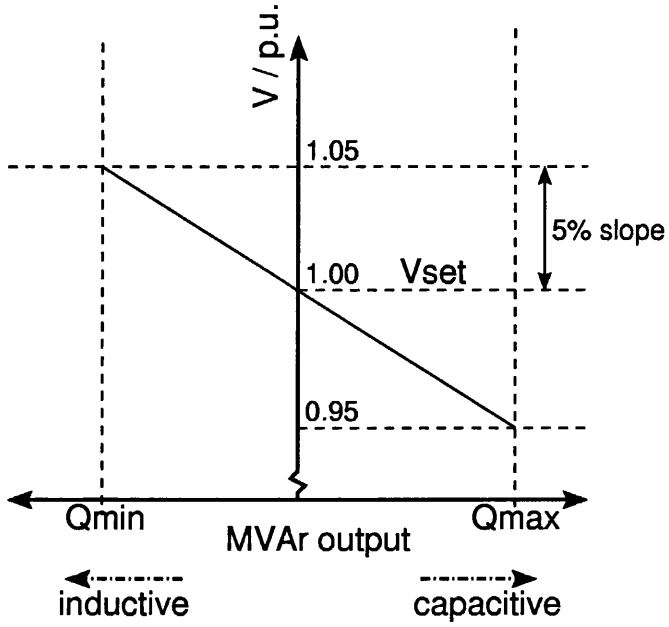

Figure 2.1: Example of a typical voltage-VAr characteristic of a voltage controlled compensation devices

2.3.2 Second generation compensation devices

The *second generation* devices developed as a result of the invention of the thyristor. Thyristors are silicon devices with a PNPN structure that act in a similar manner to diodes, except that they will only conduct when a gate current is applied. Once a gate current is applied the devices will continuously conduct until the field voltage returns to zero, when they will switch off.

By connecting two back to back thyristors in parallel – as a single device this is called a *Triac* – in series with a reactor, as shown in figure 2.2, a circuit can be formed which exhibits a variable reactive characteristic.

For each half cycle, one of the thyristors can be switched on, or *fired*, allowing

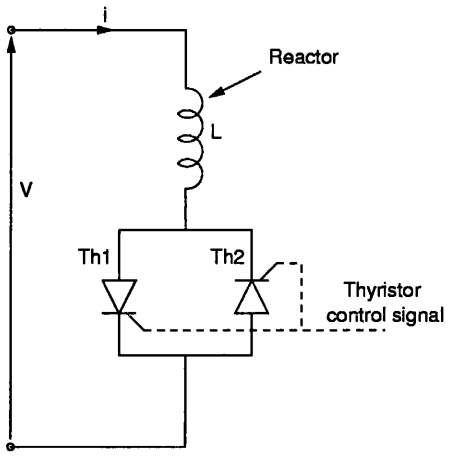

Figure 2.2: Simple controlled susceptance circuit formed by two thyristors and a reactor

a measured fraction of the current to pass through. By adjusting the delay between the beginning of the cycle and the point at which the thyristor is switched on the apparent reactance of the circuit can be changed. The phase difference between the start of the cycle and firing the thyristor is referred to as the *firing angle*. Table 2.2 lists compensation devices utilising thyristor technology.

The thyristor switched capacitor uses a thyristor circuit to perform the function of a mechanical switch. These devices can be combined with thyristor controlled reactors – and other supporting devices – to make a device called a *Static VAr Compensator* (SVC). This revolutionary device provides rapid, continuously variable, shunt reactive power compensation [Byerly *et al*, 1982]. SVCs are actually a family of devices, and their various configurations and technologies will be explored later in this section. The main objectives of the installation of SVCs on a bulk power system are detailed in the following list [Hauth *et al*, 1982].

Table 2.2: Second generation compensation devices

Device name	Abbreviation
Thyristor controlled reactor	TCR
Thyristor switched capacitor	TSC
Static VAr compensator	SVC
Thyristor switched series compensator	TSSC / TSSR
Thyristor controlled series compensator	TCSC / TCSR
Thyristor controlled braking resistor	TCBR
Thyristor controlled phase shifting transformer	TCPST
Line commutated converter compensator	LCC

Objectives of SVC siting on power systems

- Reactive power flow control during the steady state to:
  - minimise excessive system losses;
  - maintain desired voltage profile on transmission network.
- Control voltage variation due to:
  - daily load cycle;
  - repetitive impact loads such as arc furnaces (voltage flicker);
  - synchronising power flow swings;
  - dynamic variations in high voltage DC converters;
  - load rejection (when supply is cut to a load to maintain transmission system power balance).
- Power system stability improvement to:
  - maintain steady state power transfer capacity;
  - prevent transient instability;

- prevent voltage instability or voltage collapse;
- prevent oscillatory dynamic instability.

Classification of SVC plant

There are a number of design characteristics for the classification of SVC plant. The control of the output can be by active feedback or entirely passive; for example, the inherent characteristic of the static components such as saturable reactors. In addition, switching can be achieved via thyristors or conventional (non-solid state) switches [Hauth *et al*, 1982]. The contemporary design for an SVC, as shown in figure 2.3 can be defined as having some form of active control – voltage regulator feed-back control for example – and one or more thyristors switch assemblies to control output.

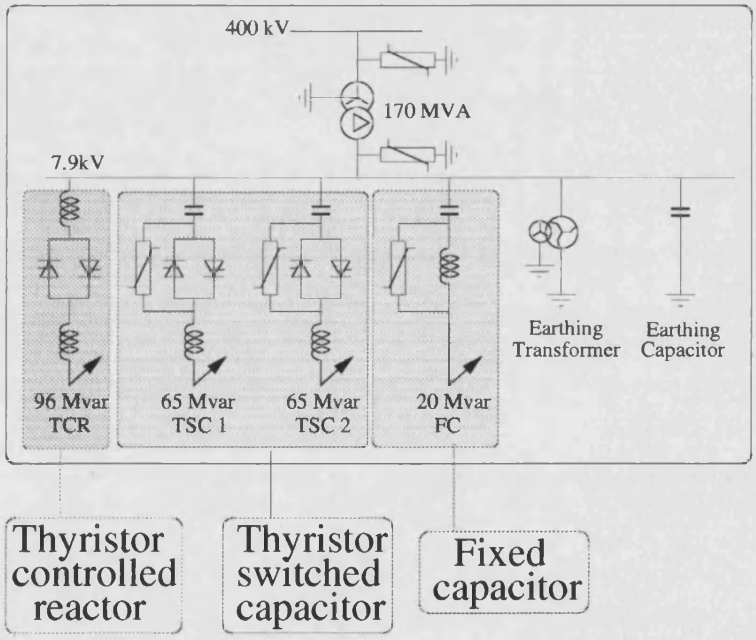

Figure 2.3: Layout of a typical SVC

When operating in the linear region of the steady state control characteristic (see figure 2.4) the thyristors control the current through the SVC reactors, and thyristors switch the SVC capacitor banks. In terms of the net reactive

output, switching of the capacitors gives coarse-grained control of the output and adjusting the firing angle of the thyristors gives fine-grained control. Assuming that the inductor is rated at an equal level to a single capacitor, any level between the maximum capacitive and maximum inductive VAr output can be achieved. Fixed capacitor assemblies are also present to perform harmonic filtering.

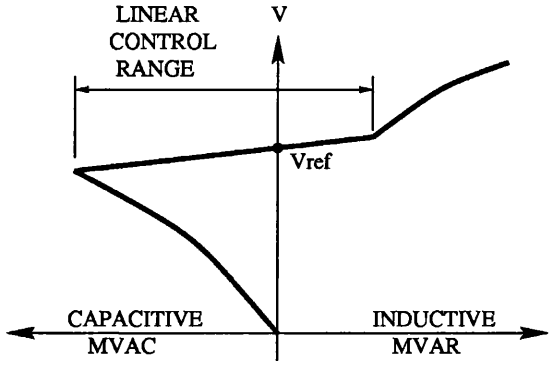

Figure 2.4: Typical SVC control characteristic

There are a number of alternative SVC configurations. Byerly *et al* [1982] categorised the different configurations as follows.

- Thyristor-controlled reactor (TCR), fixed capacitor (FC)

  Positive VArs are supplied by two or more fixed capacitor banks. The TCR is generally rated larger than the total fixed capacitance to compensate (or cancel) this capacitance and provide net negative VArs. These devices are associated with a high generation of harmonics, requiring extensive filtering schemes, and high losses. Due to their simplicity and modularity, they offer flexibility in control and up-rating. Excessive short time overloading – providing VArs outside its normal operating range – is not an inherent feature.

- Segmented TCR-FC

  Components, operation and performance are identical to the straight TCR-FC but the single large TCR is replaced by two or more smaller

segments. This results in a much reduced harmonic content in the output, but suffers from increased cost and reduced efficiency.

- 12-pulse TCR-FC

  Harmonic characteristics of the segmented TCR-FC can be further improved by the use of the *twelve pulse* configuration. By using two coupling transformers, or one with two secondary windings (one wye connected, the other delta), and dividing the reactive elements equally between these windings, a 30 ° phase shift is created between the outputs of the two halves. This results in an effective cancellation of all but characteristic twelfth harmonics.

- High impedance thyristor controlled transformer (TCT)

  Controlled reactance is provided by the impedance of a specially designed transformer instead of air core reactors. The transformer is designed with a leakage impedance of approximately 100%, and a delta connected thyristor on the secondary winding controls the flow of short circuit current through this impedance. This gives the device an inherent overload capability – during severe transient over-voltages the TCT has the capability for excessive short term VAr absorption. This type of device is associated with a high capital cost.

- Thyristor switched capacitor (TSC), TCR

  Very similar to the TCR-FC fixed scheme but the capacitor banks are in series with a solid state switch. The rating of the reactor bank is only a fraction of the total output, and the amount of capacitance is changed in discrete steps so as to keep the operation of the reactor bank within its normal control range. As well as reduced operating losses this type of device has improved performance during large system disturbance – when the demand for compensation exceeds the linear control range of the SVC. During such disturbances, "fixed" capacitor type compensators

act as parallel LC circuits, and oscillations can be established between the system and the LC circuit. Since the capacitor banks in the TSC-TCR can be switched in and out rapidly with minimal disturbance to the system, oscillations can be avoided. They suffer the disadvantage of increased plant cost and control complexity. Horwill *et al* [1991] presented a detailed description of the design of two SVCs of this type, sited in the south of the UK. They pay particular attention to how the SVCs have been individually tailored to fit the requirements of each specific site.

- Mechanically switched capacitors (MSC), TCR

  Provides many of the performance features of the TSC-TCR, but at a lower capital cost. Conventional switches are used instead of thyristors to control current through the capacitors. At best, the mechanical switching can be accomplished in four cycles compared to between a half and one cycle for thyristor switches. This slower response and the "life" of mechanical switches precludes their use for steady-state voltage regulation.

- Saturable reactor (SR)

  The saturable reactor VAr compensator does not employ any solid state switches or active control. The SR is a self regulating device that responds to changes in its terminal voltage. The voltage regulation characteristics of the SR compensator is dependent upon the natural saturation characteristics of the iron-core reactor. SR type compensators have the best harmonic character of any commercially available SVC. Series slope-correction capacitors are required to modify the voltage regulation characteristics (see figure 2.1), and these slow the response time to a level comparable to solid-state SVCs, and introduce harmonic effects. They are also far less flexible in terms of control and up-rating. Changing the voltage reference point – in this case the saturation point – requires load-tap changing coupling transformers.

All things considered, SRs are of a similar cost to solid-state devices, and are more lossy than switched capacitor devices as at zero net VAr output there are still losses in the coupling transformer of the reactor itself.

Analysis of the costs of the different configurations reveals several factors contributing to the economics. In respect of the capital cost the most expensive components are the solid state devices, non-conventional reactor and non-conventional transformer designs, and HV and EHV harmonic filter banks. In addition to this, Byerly *et al* [1982] suggested that operational and maintenance costs can be of greater importance in such analyses (the difference in losses between various SVC configuration are significant).

Specific investigations into the performance of SVCs have been outlined by Cañizares and Faur [1999], who presented the details of steady-state models used in analyses of the effects of SVCs on voltage collapse. On a similar theme, EL-Sadek *et al* [1998b] presented an investigation into the use of SVCs combined with series capacitors for enhancement of steady-state voltage stabilities. They conclude that a improvement can be attained with a reduced capital cost, although there is a risk of sub-synchronous resonance. Recently, Thukaram and Lomi [2000] summarised the various methods for assessing voltage stability, and presented a analysis method using normal load flow solutions. They demonstrated the use of this measure to assist in selecting the optimal size of an SVC, to be installed at a certain node.

Finally, an extensive bibliography of compensation literature, up to 1982, was assembled by Gavrilovic *et al* [1983], and a description of a technique for modelling SVCs is provided by Hiskens and Hill [1992].

Although these second generation devices are flexible and fast, the fast switching of the current produces severe harmonics, necessitating the installation of sizable and costly filtering equipment. These problems have been addressed by the third generation of devices.

2.3.3 Third generation compensation devices

The *third generation* devices signify a fundamental change in the design of reactive power compensation. These devices are given in table 2.3.

Table 2.3: Third generation compensation devices

Device name	Abbreviation
Static synchronous compensator	STATCOM
Solid-state series compensator	SSSC
Unified power flow converter	UPFC
Self-commutated converter compensator	SCC

The limitation in the development of second generation devices was caused by the fact that once a thyristor has been fired there is no way to turn it off, except by waiting for the voltage to return to zero. This alone limits the operating speed to twice the frequency of the system voltage: 100Hz in the UK, for example. Third generation devices resulted from the availability, in the early nineties, of *Gate Turn Off* (GTO) thyristors that could operate at high power ratings. Third generation compensation devices have more in common with DSP devices than with their first generation relatives.

The basic building block of these devices is the *Solid-State Synchronous Voltage Source* (SVS), which is analogous to a rotating synchronous condenser in its ability to maintain capacitive output at low system voltages and failure to produce harmonic resonances. However, due to its solid-state nature it does not suffer the mechanical drawback such as inertia limited response, potential for rotational instability, low short circuit impedance and high maintenance cost. Furthermore, it can dynamically exchange *real power* with the AC system, provided it is coupled to a suitable energy source to absorb or supply the power it absorbs from, or supplies to, the AC system. The generalised structure of a SVS is shown in figure 2.5.

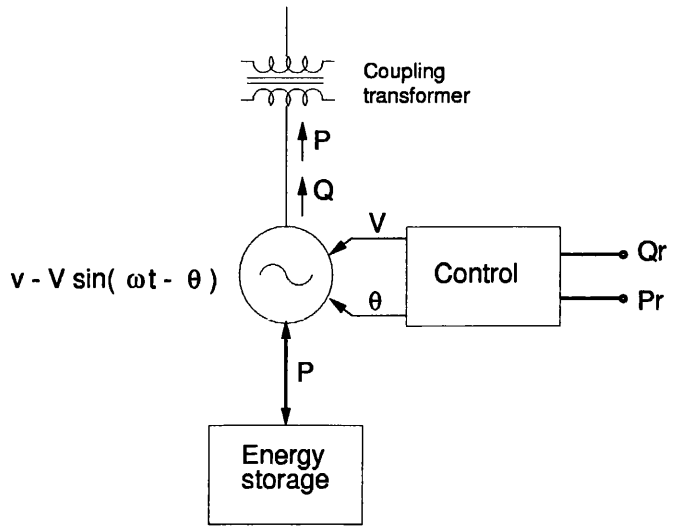

Figure 2.5: Generalised synchronous voltage source

The control device uses reference inputs Qr and Pr to produce the control signals for voltage (V) and phase angle (θ), which in turn define the output voltage of the SVS. When the voltage source is greater than the line voltage, leading reactive current is drawn from the line and the equipment appears as a capacitor. When the voltage source is smaller than the line voltage, lagging reactive current is drawn. Real power may be exchanged by adjusting the phase of generated voltage, the SVS generating power when leading the line voltage and absorbing power when lagging it. In practise a small amount of real power is also drawn from the line to supply the loss of the converter

The SVS voltage source is implemented by a *switching power converter* which interfaces the DC energy storage and the AC transmission system. The contemporary version of the SVS switching power converter, the *Voltage Sourced Converter* (VSC)– or *Voltage Sourced Inverter* (VSI)– is shown in figure 2.6. The figure shows the most basic arrangement for a device of this type. Each phase is supplied with a two level square wave, shifted in phase by 120 °, producing three-level line to line voltages. The discontinuities in the waveforms can be reduced by electro-magnetically combining multiple converters of slightly

different amplitudes and voltages, to produce a smoother stepped AC signal at the outputs.

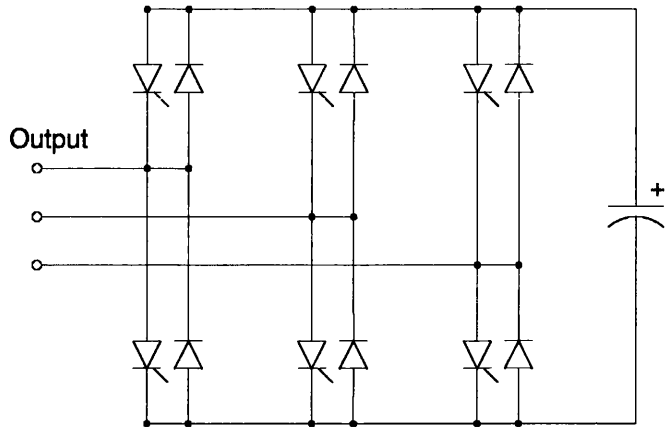

Figure 2.6: Basic six-pulse two-level Voltage Source Converter

The STATCON, or STATCOM, is a VSC connected to a busbar through an inductance – from the coupling transformer – to supply shunt reactive compensation. Note that the name STATCON was the first acronym used to describe such devices, taking its name from *STATic CONdenser*, because the steady state output characteristics are similar to those of the rotating synchronous condenser. The name STATCOM comes from *STATic Synchronous COMpensator* and is the accepted term within the IEEE, CIGRE, National Grid and most manufactures. The expression *Advanced Static VAr Compensator* (ASVC) and *Self-Commutated Static VAr Compensator* (SCC) may also be seen.

The STATCOM outperforms the SVC in a number of ways, as listed below.

- Delivers VArs over its full range regardless of the system voltage magnitude.
- Has a 20% short term overload capability.
- Requires 15-35% less MVA rating to deliver the same level of support for:
  - system steady-state power transfer;
  - dynamic voltage support;

  - transient stability performance.

- Order of magnitude better performance than SVC in:
  - damping system oscillations;
  - unbalanced operation;
  - flicker reduction.
- Much smaller environmental footprint than an equivalent SVC.
- Does not:
  - contribute to system resonances;
  - introduce resonances;
  - add to fault level other than rating;
  - have electromechanical oscillations.
- Can act as a voltage source in starting other converter equipment.
- Can be used together with other types of compensation equipment.
- Does not require harmonic filters.

The *Solid-State Series Compensator* (SSSC) is a VSC connected in series to a line through a coupling transformer. It is a very powerful tool for controlling power flow, able to control the angle and impedance of the line, and its real power exchange capability makes it very effective in improving dynamic stability.

The *Unified power flow converter* (UPFC) is a combination of a STATCOM and a SSSC which share the same energy storage device. This configuration allows the UPFC to function as a separate STATCOM and SSSC or a double rated STATCOM.

Further details of modern reactive compensation devices can be found in the following references: [Erinmez and Foss, 1998; Gyugyi, 1994; Schauder *et al*, 1996].

2.4 Tap-changing transformers

Another method for voltage control is in the use of tap-changing transformers [Weedy, 1987]. By changing the tap setting of the device, the turns ratio of the transformer can be changed. The voltage change between successive taps is often 1.25 per cent of the nominal voltage. This small change is necessary to avoid large disturbances at consumer busbars.

One form of tap-changing transformer is the automatic tap-changing transformer. This device performs voltage regulation and has been shown to be useful for improving voltage stability for both steady and transient states [EL-Sadek *et al*, 1998a]. It automatically adjusts its tap settings in sympathy with the voltage at one of the busbars to which its connected. This busbar is referred to as the *control busbar*. The control busbar voltage is compared to a reference voltage, at which the tap is set to the nominal value. If the voltage is below the reference, the tap setting is increased which will act to increase the voltage at the control busbar. Similarly, below reference control busbar voltages cause the tap setting to decrease.

2.5 Placement of compensation devices

In section 2.2, the case was made for the ongoing need to site compensation devices on a transmission system to maintain the quality and security of the power supply. A number of solutions to the reactive power compensation problem have been presented in section 2.3. As discussed, there is great

variance in the sophistication of the various devices, which goes hand in hand with a similar variance in costs. To give an idea of scale, a 150 MVAr SVC costs between eight and ten million pounds to implement, and would have a footprint larger than a football field [Erinmez, 2000]. This highlights two problems: firstly that the large capital cost restricts the number of such devices that can economically be purchased, and secondly that the size restricts the set of candidate locations at which an SVC could suitably be sited. In addition, the objectives of SVC siting have direct commercial implications. For example, although system losses can exist without affecting the quality or security of a system they are a financial burden to the system operator. The compensation siting problem is therefore one of optimisation. A plan for the siting of compensation devices must be devised that complies with the restrictions on location and size, satisfies the aims for security and quality, while achieving optimum operational performance. This problem is defined and discussed in more detail in chapter 3.

Chapter Three

Compensation siting techniques

3.1 Problem outline

The premise of the reactive power compensation planning problem is a simple one: *define the specification for the installation of new reactive sources which achieves the maximum benefit at the least possible cost, while satisfying system and operational constraints.* This statement encompasses the full scope of the problem:

- *define the specification for the installation of new reactive sources.* The specification needs to define the types of compensation to be installed, the parameters of the compensation devices – their size and any control parameters – and their locations.
- *maximum benefit.* Sections 2.2 and 2.3 detailed the benefits of reactive power compensation. A power system planner needs to formulate measures to evaluate the benefits of a particular siting plan. Due to the massive scope of the effects that modifications to a power system can produce, planning algorithms presented within the literature, as will be investigated later in this chapter, generally focus on a certain subset of effects. For example, in planning exercises, only steady state effects are usually considered. This can greatly reduce the problem space of the planning exercise because, for example, the response speed of the device can be ignored, and no costly dynamic simulation of the network is necessary.
- *least cost.* Costing is very much part of achieving "maximum benefit." The choice of how a solution is costed is mainly down to the system

operator: the capital cost of installing new sources is a cost, but factors relating to system losses and voltages can also be considered as an indirect cost. In practice, if all the effects considered can be interpreted as costs they can then be added together to give one value for the measure of a system's performance. If no such biasing can be formulated – for example, if an operator does not wish to decide on a fixed "pounds per volt of voltage deviation" value – more sophisticated multi-objective techniques need to be used to compare competing siting plans. Such multi-objective problems are discussed in sections 4.3 and 4.4 in greater detail.

- *satisfying all constraints.* The main constraints can be categorised as *expansion constraints* – the physical limitations in the opportunities for the placement of devices – and *operational constraints* – power flow balance; line flow limits; voltage magnitude limits; limits in voltage phase angle difference; transformer tap limits; real and reactive power generation limits; and reactive power compensation limits.

Fundamentally, the formulation of the problem is down to the discretion of the power system operator: the factors they consider important and the bias they place on those factors.

3.1.1 The planning and operational problems

As the problem is considered in greater detail, it becomes apparent that a solution to the problem needs to include more than just the plan for the placement of devices. The problem can be thought of as being made of two sub-problems: firstly, finding the optimum siting plan and, secondly, finding out how to optimise the system to get maximum benefit from the siting plan. The reason for this second problem can be explained as follows. Imagine that within the placement software a candidate solution is presented. Within

the candidate solution a scheme for the installation of new VAr sources is described. The question the algorithm then needs to answer is "how well will this scheme perform?" To do this, it could just install this new scheme on the transmission system and examine the operational characteristic via load flow. The problem is that in the real world a power system is not just left to run on its own and there are a number of parameters that can be adjusted to maximise performance. So, to fully evaluate the performance of the new scheme, the software must also optimise these parameters. In siting software, these additional parameters, which involve things like tap settings and controller characteristics, can be thought of as one part of the candidate solution, or as a separate issue to be dealt with separately. To indicate the fact that the problem is not just about *siting* compensation, the word *planning* will be used. Hence, the expression *Reactive power Compensation Planning* (RCP) will be used to refer to the problem in its entirety. The detailed formulation of the RCP problem considered in this work, including these additional parameters, is discussed in chapter 5.

In this work, a number of formulations is considered. For the purpose of performance comparisons, the formulation of the objective function will be taken from the literature, enabling the algorithm presented in this work to be compared with other contemporary techniques. One example in particular will be that of SCORPION (see section 3.5) a reative power investment package currently used by National Grid. Another formulation used in this work is an extended formulation, based on the same fundamental concepts as the SCORPION objective, but expanded to encompass optimisation of the voltage profile.

3.2 Overview of compensation placement techniques

As discussed, reactive power compensation is essential to maintain the quality, reliability and security of the transmission system. Unfortunately, installing compensation is an expensive solution: for example, a 150 MVAr SVC can cost between eight and ten million pounds [Erinmez, 2000]. Badly formulated siting plans can be highly sub-optimal, and the large quantities of money associated make the issue of optimal placement very important for power system operators. In addition, the size of the problem space is proportional to the size of the transmission system and the number of contingent states considered. The problem space is also non differentiable, discrete, nonlinear and multi-modal [Pilgrim *et al*, 1999].

As one of the most challenging optimisation problems in power systems, the RCP problem has always received attention from the cutting edge of optimisation algorithms. This section details the various flavours of algorithm that can be found in the literature.

3.2.1 SVC siting techniques

Many optimisation techniques have been used to solve VAr siting problems. The following sections describe the most commonly used algorithms, sorted in respect to the date at which they start to appear in the literature.

Algebraic techniques

The first paper describing a structured approach to the placement of reactive compensation was that of Cook [1959]. The author points out the pitfalls of the then existing *ad hoc* techniques: an improperly located fixed capacitor can

actually increase the energy loss in a line. Cook then presents an analysis of the effects of fixed capacitors on radial circuits with distributed loads, taking into account the effects of a periodic load cycle. A set of curves are presented describing the optimum positions and most economical ratings of fixed capacitors on a radial circuit. In 1961, Cook extended the theory to include both fixed and switched capacitors [Cook, 1961]. Cook's analytical technique is satisfactory for an individual circuit, but problems arise when applying such techniques to a network: every time extra compensation is installed on a line the profile of the network changes, and previous sitings need to reevaluated to take account of this change.

Gradient techniques

Heydt and Grady [1975] presented a technique for optimal VAr sitting. The load flow problem is formulated in a linearised form and a "gradient scheme" is used to steer the optimisation. This basically means that a linear approximation to the problem space is formulated, with the algorithm performing hill climbing to optimise the control variables: the amount of compensation to be added at each busbar.

Hill climbing algorithms, or gradient algorithms, work by examining the local area surrounding solutions for better solutions. This is done by making small changes to the solution and reevaluating its performance. Any changes that produce an improvement in performance are kept. Although extremely fast and simple to implement, they are problematic because they can only find improved solutions while one exists in the local area and can, therefore, easily get stuck in local minima or maxima during minimisation or maximisation, respectively.

Linear Programming

Happ and Wirgau [1978] presented a method for planning reactive compensation, notably including the "...newly developed variable shunt reactive control devices referred to here as Static VAr Control devices (SVCs)." The algorithm aims to determine the minimum amount of additional reactive power required to maintain the busbar voltages within acceptable levels. The algorithm uses a *Linear Programming* (LP) technique to optimise the control variables. Solutions are validated by static, that is, steady state, and dynamic modelling. They also presented a model for simulating a thyristor-controlled static VAr device, which they then use in conjunction with a linear programming technique.

LP techniques find an optimal solution to a linear representation of a problem. A solution to a problem is a set of quantified variables (also called decision variables). These variables are the unknown quantities that must be evaluated at the optima. Three sets of linear equations, in terms of these variables, are then required for the formulation of the LP problem: objective functions, constraints and variable bounds. Objective functions are mathematical expressions of the goals of the optimisation, constraints combine the variables to express limits on possible solutions, and variable bounds express the limits on the values that variables may take.

These equations combine to form a problem space – that is, a hyper-space in which each dimension is a variable – that has two regions: feasible and infeasible. A point in the infeasible region, unlike a point in the feasible region, violates at least one constraint or variable bound. Within this problem space there exist flat surfaces, or hyper-planes, such that each point on the plane is associated with the same objective function value; one could imagine these as hyper-contours of the objective function. The optimum solutions are at the points that lie on the best possible objective function hyper-contour and that also exist within the feasible region.

Due to the fact that the problem has been formulated from linear equations, two important aspects of LP theory can be raised. Firstly, as the objective function must increase as it moves away to infinity (in some direction) it can be shown that this optimum intersection lies on the very edge of the feasible region. Secondly, because the feasible region is bounded by flat planes, any other flat surface touching the edge of this region must be touching it at the corner-points. Any other points of contact, caused by two or more corner-points being optimum so that the *edge* connecting them must also lie on the same hyper-contour, must have the same objective function value. Therefore, *an* optimum solution to the problem must be one of the corner-points. The LP algorithm, therefore, only has to check these points to find the optima; in fact it does not even have to check this many. These concepts are illustrated in figure 3.1.

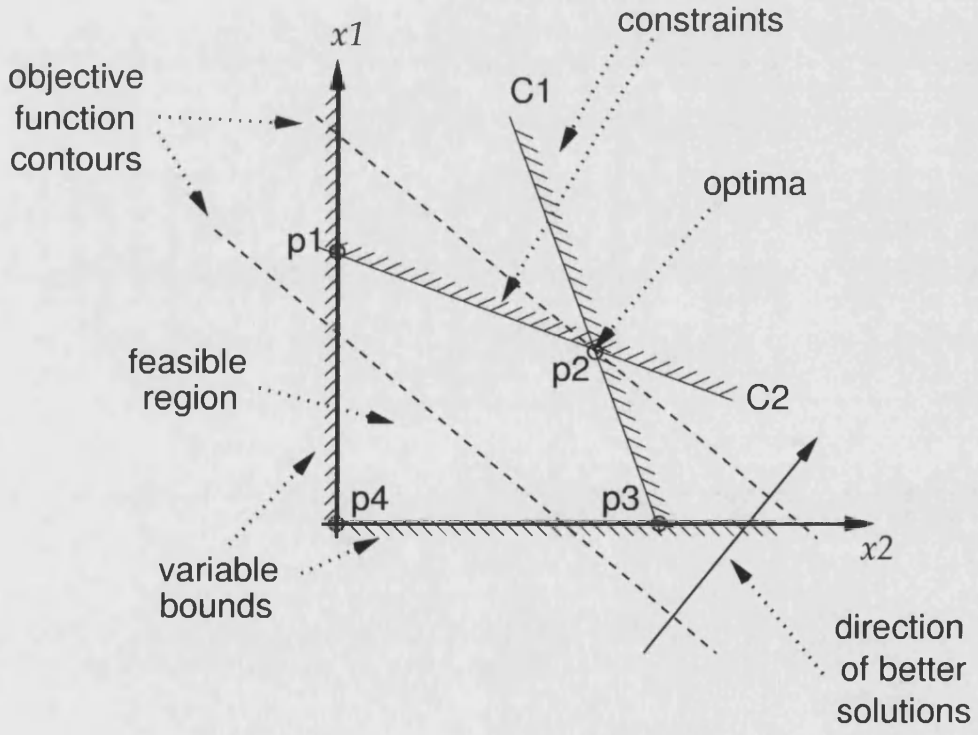

Figure 3.1: Example of a typical linear programming problem space

The figure depicts the LP problem space of a simple fictitious example in which there are only two variables, $x1$ and $x2$. This means that the problem space is

two dimensional and the aforementioned hyper-planes are one dimensional: simply, lines. In this example the variables must be positive and there are two constraint equations; the feasible region is, therefore, bounded by four linear equations with respect to $x1$ and $x2$. These are the constraints functions, $C1$ and $C2$, and the variable bounds, which lie on the two axes. Note that infeasible points lie on the shaded side of these lines and all such contour lines actually go to infinity in both directions. Two examples of objective function contours are shown by the dashed lines. The objective function value for all points on a line are the same. Note that these lines are parallel and the value of the objective function increases – that is, the solutions gets better – as one considers points in the direction shown by the thin solid arrow.

As can be seen, there are four corner-points of the feasible region, p1, p2, p3 and p4. As the perimeter of the feasible region is composed of straight lines and the objective function is linear, there must be either a single optimum solution at one of the points p1, p2, p3 or p4, or a set of optimum values that fall on one of the lines connecting the corner-points: p1 to p2, p2 to p3, p3 to p4 or p4 to p1. In either case, checking the objective function at the four points will reveal at least one solution that is optimal. In this example, the optimal solution is the one represented by the point p2.

LP is a powerful technique that has been successfully applied to many engineering problems. It does, however, have two weaknesses when applied to the RCP problem. Firstly, is the fact that all variables are treated as being continuously variable (or *non-discrete*). Whilst being an asset for many applications, this is quite the opposite when considering the RCP problem that has no continuous variables. For example, compensation devices are only available in certain sizes and variable transformer taps can only assume a discrete set of values.

To overcome this problem there have been a number of algorithms developed based on LP that handle such discrete problems better. These include 0-1

programming and integer programming. 0-1 programming uses additional variables that can have the value of zero or one to act as "on-off" switches for other variables, an example of which would be to control whether a device were installed or not. Integer programming (and mixed integer programming) aims to fully support integer values by keeping the integer values constant and performing LP on any continuous variables. One of the integer values is adjusted by plus/minus one before repeating the LP optimisation. As this recursive optimisation is extremely time consuming further approximations are introduced to speed up optimisation [Aoki *et al*, 1988].

The final drawback to all LP based techniques that although an optimum solution can often be found quickly it is an optimum solution to an *approximation* of the problem; the performance of such a solution when applied to the original problem may, or may not, be near optimal.

LP based algorithms have continued to receive attention: Opoku [1990] used a duplex simplex linear programming technique coupled with relaxation and contingency analysis to solve the RCP problem. The method is applied to the IEEE 30-bus and 118-bus systems. Opoku outlines the modification necessary to promote convergence in LP algorithms when dealing with large, highly stressed or contingent systems. Also, Ahmed *et al* [1999] use a LP approach for coordinated optimisation of preventive transformer taps and VAr installation.

Decomposition

Lee *et al* [1986] divided the planning problem into a master problem and two sub-problems. The master problem determines the investment in reactive power compensation devices, and is particularly interesting because it presents this investment as the investment to be made for each year of the planning horizon. This is in contrast with the other planning algorithms which have been presented in the literature which assume that all compensation will be installed at the same time. As this would not necessarily be the case – maybe

the system operators can only afford one device per year – the order in which devices should be installed, and so on, becomes important. This algorithm presents the optimal solution as the siting plan for each year. The two sub-problems determine the optimal operation of the power system under normal conditions in respect to real and reactive power. Although an extremely interesting work, one must be aware that the more objectives an optimisation algorithm is given, the more compromises the algorithm may have to make. This implies that the algorithm may choose a solution that provides better *incremental* performance at the sacrifice of *final* performance: the system at the planning horizon – that is, once all the yearly investments have been made.

Lie and Deng [1997] also used a decomposition approach, but aimed specifically at the allocation of FACTS devices: in particular, variable series capacitors and static phase shifters. They present results for a variety of systems up to and including a thirty busbar system.

Recently, there have been many innovative techniques developed in the field of SVC expansion planning. Notably, most modern algorithms utilise some form of artificial intelligence.

Simulated Annealing

Simulated Annealing (SA) exploits the resemblance between a minimisation process and the cooling of molten metal: as the metal cools, the molecules are rearranged such that the potential energy of the molecular structure (of the bonds) of the metal is minimal when the cooling process is finished. The technique basically operates as a gradient algorithm: solutions making small downhill movements, but on top of this process is a measure of temperature. The temperature starts "high" and decreases, at each iteration, ending up at some lower "cool" level. At each iteration, instead of just allowing points to move downhill, there is a random chance it will allow an uphill movement to be made. The chance of such a movement being allowed to be made is

proportional to the temperature. This means that initially the search point will be fairly unconstrained and free to move around the problem space. As the temperature drops the search point will start to settle into minima, but will still be able to get out of any local minima. When the temperature has reached its minimum the search point will no longer be able to move uphill, and the search will concentrate on finding the optimal point of the minimum it is in [Jwo *et al*, 1995; Chen and Liu, 1994].

Chang and Huang [1998] formulated a scheme for optimal multi-objective SVC planning. Using parallel simulated annealing and a Lagrange multiplier, the technique is concerned with optimising for voltage stability enhancement. They defined a fuzzy performance index which represents system reactive power margin, system I^2R losses and voltage depressions at critical points, thus converting the multi-objective problem into a single objective one. The compensation siting is constrained in terms of the SVC controller characteristics, the number of SVCs and the overall capacity to be installed. The technique is tested on the IEEE 14-bus system, under normal and contingent conditions.

Chen and Liu [1994] worked on a similar optimisation using a goal attainment method based on simulated annealing: optimisation was done in terms of active power loss reduction, minimisation of SVC investment cost, system security margin robustness and reduction of the voltage deviation of the system. They discuss in detail the formulation of many of the main objectives and constraints commonly involved in an SVC placement problem. The proposed technique is tested on the AEP 14-busbar system.

Jwo *et al* [1995] used simulated annealing in conjunction with a hybrid expert system. The expert system is used to reduce the problem size by removing busbars from the list of candidate busbars. Busbars are removed if, after a unit of compensation is sited at that point, there is no improvement in the voltage profile. This expert system is also used to provide the SA stage with an initial solution from which to start optimisation. It solves the problem of having a

multi-objective objective function by replacing each objective function with a fuzzy goal. These fuzzy goals can then be combined, thus transforming the multi-objective optimisation into a single minimisation or maximisation optimisation. The technique is tested using a modification of the IEEE 30-bus system. In comparison with SA, a slight cost reduction and significant CPU time reduction are demonstrated.

Tabu search

A similar search technique to SA is the *tabu search* algorithm [Nara *et al*, 2001; Mori, 2001], which is also a extended gradient technique. The algorithm finds optimal solutions by moving the search point to a better solution in the neighbourhood of the current search point, as does a simple hill climber. The advancement of the tabu search can be found in its use of a *tabu list*, which stores all the solutions previously found, and does not allow the search point to revisit these points. The search point is, therefore, permitted to move uphill, if the only solutions that exist in the current neighbourhood are worse. Due to this controlled uphill movement, the search point may move out of local optima, and, therefore, the algorithm has the ability to find optimal solution in multi-modal problem spaces.

Evolutionary Algorithms

The field of evolutionary algorithms began with the invention of the *Genetic Algorithm* (GA) by Holland [1975]. GAs, the search algorithm based on the mechanics of natural selection and genetics, are discussed in greater depth in section 4. The growing trend over recent years has been the application of evolutionary algorithm based techniques to the siting problem. Iba [1993] presents a siting technique using GAs. The GA uses an inter-breeding/crossover algorithm that utilises system topology and objective

function heuristics, in addition to expert-system stochastic if-then rules. The technique is applied to practical 51-bus and 224-bus systems.

Two papers have been presented by Lai and Ma [1998, 1997b] utilising an evolutionary based technique which they call *Evolutionary Programming*. The technique optimises for total energy loss cost and VAr installation cost, and is applied to the IEEE 30-bus system. The two objectives are combined using a weighting method, which enables them to use a simple fitness proportionate selection scheme; see section 4.4 for a detailed discussion of multi-objective approaches and GA selection techniques. In their first publication [Lai and Ma, 1998] they used a GA with an adaptive mutation rate, the performance of which is compared to the popular LP technique called *Broyden's method*. The technique starts by using a high mutation rate and reduces it as the solution converges. The effect of this is to provoke greater initial exploration and then progressively encourage greater exploration. The technique used measures the population's convergence by comparing the minimum and maximum fitness. This technique is therefore suitable for problems where there is a near-zero chance of infeasible, or extremely bad, solutions being routinely generated from the converged population, as such solutions would confuse the monitoring of convergence. Mutation, and the choice of mutation rate are discussed in section 4.1.1 and 4.6.3 respectively. Their results strongly support the argument for the superiority of evolutionary approaches over LP techniques, citing up to a fifty percent improvement in solution cost.

Their second work [Lai and Ma, 1997b] also considers optimisation of network contingencies. One drawback of their approach is that the VAr requirement for each contingent system state is optimised *separately*; each state therefore requires a different set of new reactive sources. They suggest no way of producing a single plan that is near optimal for all cases. If all the sources required for all cases were installed then the results would be suboptimal.

Lee and Yang [1998] analyse the effectiveness of the three main evolutionary

algorithms: evolutionary programming, evolutionary strategy, and genetic algorithms. Whilst based on the genetic algorithm, the evolutionary programming and evolutionary strategy approaches do not use a crossover operator, and instead rely solely on mutation to generate diversity. They bench test these against a linear programming technique. Optimisation is carried out to minimise fuel consumption for the IEEE 30-busbar test system. They conclude by stating that the three evolutionary algorithms exhibit very similar performance characteristics, and all out-perform the linear programming approach. They also discuss the effect of different mutation rates: lower rates providing faster convergence but a greater chance of converging on local optima.

There have been several studies concerned with the problem of siting capacitors on distribution systems: the areas of the network between the Supergrid substations and the customer. Abdullah *et al* [1997] investigated the use of GAs for such problems. Although successful results are presented, unfortunately, because results are only presented for a test system of 5 busbars, strong conclusions about the performance of GAs when applied to real-world sized problems cannot readily be drawn. Similarly, Chung and Leung [1999] presented the results of capacitor siting on a nine busbar circuit. Their algorithm attempts to minimise the cost of compensation under the limits of harmonic distortion: a rarely considered constraint. Capacitor placement using GAs has also been implemented by Kalyuzhny *et al* [2000] who paid particular attention to the dynamic performance of the system, and Delfanti *et al* [2000] who presented results for single case studies of the 500 busbar Italian system using a combination of LP and GA techniques.

Lee *et al* [1995] and, recently, Mantovani *et al* [2001] both presented techniques using GAs in combination with LP to solve the operational sub-problem. Mantovani *et al* used a sub-problem based approach, using GA to optimise the discrete aspects of the problem (transformer taps and capacitor allocation) and a LP algorithm to optimise the continuous variables of the system operating

state. They presented results for the 30 busbar IEEE system and a 309 busbar real system, nether with contingencies. Their algorithm uses integer coded GA with a proportional selection method. A single objective function is formulated from VAr installation costs and a mismatch penalty. Their results concentrate on investigating the effects of varying the parameters of the GA: population size, mutation rate, and crossover probability are all considered. They conclude that a population of three hundred and eighty, a probability of crossover of one in two, and a probability of mutation of one in ten thousand produced the best results. See chapter 4, and in particular section 4.6, for a discussion of these factors.

3.2.2 SVC placement using critical modes of voltage collapse

Mansour *et al* [1994] took quite a different approach to the problem of compensation placement: theirs is not an optimisation algorithm, but an analytical technique. Placement is based on analysis of the system power flows at the point of maximum loadability, that is, the system is loaded to near its point of collapse, and modal analysis is performed on this stressed system. SVCs are then placed at busbars vulnerable to voltage collapse. Results are presented for the B.C. system in Canada.

3.3 Additional developments in RCP

In addition to the development of specific planning algorithms, there have been advances in generic aspects of the problem.

Bridenbaugh *et al* [1992] highlighted an important consideration of the siting problem: optimisation of existing plant. Any siting algorithm, from heuristic to artificial intelligence approaches, needs to evaluate the performance of

each suggested, or *candidate*, solution. They noted that to make a reliable assessment, all plant that contributes to the system performance needs to be considered. This increases the scale of the problem, and they discussed how conventional power flow techniques may no longer be sufficient for evaluating solutions. Conventional power flow calculates system conditions, such as voltage angles and power flows, from the physical values of the transmission system and the generation and demand patterns given as input. This means that the system equations are, therefore, only solved for one specific set of equipment settings.

If an adequate solution is not found the various settings – tap settings and so on – must be adjusted and the power flow re-run. This method requires heuristic knowledge of the user and as the network size increases the complexity of this problem increases. This is where *Optimal Power Flow* (OPF) software becomes invaluable. OPF is a power flow program which not only solves the power flow equations but optimally adjusts system control variables to achieve desired results. The system data is specified in much the same manner as that of conventional power flow software, but additional data is required to specify control ranges, constraints and limits. By using OPF software, the handling of these system control variables can be treated as a sub-problem of the siting problem. The optimal-siting algorithm only needs to formulate the proposed siting plan, not the complete system specification. The complete system specification is only realised while evaluating the performance of the siting plan, using the OPF software.

Aoki *et al* [1988] address the problem of the discrete aspect to the specification of VAr source installation. The problem arises from the fact that commercially available compensation devices are not available in continuous sizes [Chung and Leung, 1999]. For example, SVCs for 400 KV transmission systems typically operate between 75 MVArs inductive to 150 MVArs capacitive. Note that potentially more than one device may be installed at a single location.

This discrete aspect of the problem is important when choosing the optimisation algorithm. Some algorithms, LP for example, assume all variables are continuous. The solution the LP algorithm presents as the optimal solution must then be converted into a realisable siting plan: the standard way of doing this is to round up the sizes of the compensation devices to the next commercially available size. This can result in a suboptimal solution. To illustrate this point consider two competing siting algorithms. The first, a non-discrete approach, finds a solution involving two 10 MVAr SVCs for locations A and B; the second, a discrete approach, finds a solution involving one 100 MVAr SVC for location C. The first approach appears better until the solution is implemented and it is found that devices are available in units of 100 MVArs. The first solution would then be converted into a suboptimal solution involving two 100 MVAr SVCs. Aoki *et al* then present a technique using a *Mixed Integer Programming* algorithm which treats capacitors as discrete devices in real scale systems.

Chen *et al* [1995] proposed an algorithm for simplifying the problem space of such problems by identifying *weak buses*. The weak buses are those at which the voltage is already approaching the maximum system limit and are far more likely to be candidates for reactive compensation.

Finally, Vaahedi *et al* [1999] demonstrated that existing OPF/VAr planning tools can be accurately used to address voltage stability constrained VAr planning and voltage stability applications, in addition to the traditional feasibility criteria of acceptable voltage profile.

3.4 Real-world examples of compensation planning

Hauth *et al* [1978] published details of the "Application of a static VAr system

to regulate system voltages in Western Nebraska". The site and rating of the VAr sources were chosen via a combination of repeated load flow studies and heuristic knowledge. Also, Bridenbaugh *et al* [1992] examine the new VAr scheme for the Ohio electricity system, Mansour *et al* [1994] examine the B.C. system in Canada, and Levitin *et al* [1998] use a GA for placement on the Israeli power system. The papers describe the considerations of a utility system planner contemplating controllable shunt compensation for their systems.

3.5 A typical example of an SVC placement package: SCORPION

SCORPION [Thomas *et al*, 1995; MacQueen *et al*, 1998] is the name of the reactive power compensation placement package developed and used by National Grid, and is of note for a number of reasons. Firstly, it is an example of a fully developed solution to the compensation problem, secondly, it is being used for the planning of the England and Wales power system. In essence, when SCORPION is used (in *planning mode*), it solving exactly the same problem that this project has been angled towards, and therefore an element of validating *Genetic Compensation Placement* (GCP) will be by comparison with SCORPION.

3.5.1 Optimal base; secure contingencies

SCORPION performs reactive power compensation plant planning for the UK power system. The catch phrase for SCORPION's placement algorithm is *"optimal base; secure contingencies"*. The thinking behind this is that although contingencies happen, the system does not remain in such a state for very long. For the vast majority of time the operational state of the network is non contingent. This state of being not affected by any contingency is referred to

as the intact state (in the physical sense that nothing is broken) or the base case (referring to the data set that describes the intact system). Due to the short lived nature of contingent states, there is little to be gained, financially, by optimising the performance of the power system in these states. The important thing is that transmission can be maintained: the system is *secure for contingencies*. What is desirable, however, is that the performance is *optimal for the base case*, where issues to do with running costs become important.

This is achieved by using a linear programming approach which adds to the system's compensation until all states are secure, and then performing a backtracking stage to reduce unnecessary installation and optimise the base case while maintaining the overall security.

3.5.2 SCORPION optimisation

SCORPION optimises the pattern of reactive generation for an instantaneous state ('snapshot') of the transmission, with a specified pattern of active and reactive demand and active power generation, subject to maintaining voltages and reactive power generation within acceptable limits. This system state is likely to be the normal or intact state, but in practise most of the reactive capability of the system is not used in any one state, and is held in reserve to maintain voltages during unplanned (contingent) states, and to accommodate system conditions throughout the year. SCORPION therefore performs optimisation for a number of system states.

This optimisation is performed subject to the requirement that the system voltages be maintained within an acceptable range for normal operation. Any additional reactive generation which is installed to satisfy the constraints in one system can provide reactive support for the other system states. Each node is assumed to have two distinct types of reactive power compensation associated with it: *actual* and *potential*. Actual generators represent the existing

physical limits of the nodes generation; actual generators are assigned a low cost for usage. Each node also has associated potential generation which represents the ability to install new generation at that node, if required. Such generation is assigned a high cost for usage.

SCORPION uses three phases of operation during the optimisation:

- **Phase 1:** The system states to be considered are ranked on the basis of their separate VAr requirements. This ordering is used in subsequent phases, as described.

- **Phase 2:** Single state reactive optimisation is performed for each state individually, starting with the one with the greatest requirement and proceeding in order. Any potential generation used is costed appropriately, before being converted to existing generation, making it available for use in subsequent states.

  At the end of phase 2 sufficient reactive generation will have been installed to satisfy voltage and reactive constraints for every system state. This solution is not intended to be optimal, but to achieve security by installing sufficient amounts of compensation for each case when considered on its own; this may then result in excessive amounts of compensation being placed in total. Reducing this compensation to achieve optimality is addressed in phase three.

- **Phase 3:** This phase attempts to reduce the reactive power generation requirement without violation any of the constraints. This is achieved by using a *backtracking* technique. The algorithm proceeds as follows:

  - Select the state that had the greatest requirement, according to phase 1.
  - *The subtraction step:* remove the reactive generation added to the network after the single state optimisation performed in phase 2.

- *The addition step:* repeat the single state optimisation for those system states that would be affected by the removal step. Additional reactive generation can be installed if required.
- If another state is available from the phase 1 ordering, select this state and repeat from the subtraction step, otherwise finish.

The premise behind this backtracking procedure is that more reactive power generation is removed in the subtraction step than is subsequently added in the addition step.

At the heart of SCORPION is a linear programming optimisation engine [Cheng and MacQueen, 1996] and it therefore suffers the drawbacks that are characteristic of linear programming techniques: it does not take account of the discrete sizes of available MVAr sources nor the discrete nature of plant settings. Approximating continuous variables to the closest discrete values can have a number of effects. Firstly, adjusting transformer taps can generate infeasibility, but on planning time scales these cannot be decided accurately, and, secondly, rounding off reactive sources to the nearest available size adds cost, but does not generate infeasibility if fixed capacitors are not considered because large SVCs can give smaller output. These factors necessitate user intervention which is needed to interpret and manipulate the siting solution to produce an executable plan. Another drawback is that it does not try to optimise the type of of reactive compensation sited: for example, fixed capacitors or SVCs. Note that SCORPION has two modes of operation, planning and costing. The costing mode is used for pricing studies and financial investigation and is outside the scope of this project [Chebbo *et al*, 1999].

3.5.3 Consideration of auxiliary systems parameters

As discussed, voltage is fundamentally linked to reactive power, and, hence, with the the reactive power compensation planning problem. Therefore the planning problem must consider system parameters that relate to system voltages. These parameters are those describing variable transformer tap settings and reactive power generator voltage set points.

As variable transformer taps affect system voltages they indirectly affect the flow of reactive power. There are two categories of variable transformer taps on a power system [Ahmed *et al*, 1999]:

Preventive taps: The tap settings are fixed for all cases; they do not respond dynamically to the system state. This means that they remain the same after or during contingencies.

Corrective taps: The tap settings respond to the system state. They monitor the voltage at the busbar to which they are connected and adjust their tap settings to try to maintain the busbar voltage at some pre-configured target voltage. This means they can have a different tap setting for each system state.

These two categories of variable transformer taps present two associated categories of parameters: the tap setting for preventive taps and the target voltage for corrective taps. The target voltage for corrective taps is nominally set at one *per unit* (p.u.), that is, the actual system voltage, and do not need to be considered. Corrective tap settings, on the other hand, are considered.

As discussed in section 2.3, compensation devices with dynamic VAr output (synchronous generators and the second and third generation devices) have an operational characteristic that relates the VAr output of the device to the observed system voltage, as shown in figure 3.2.

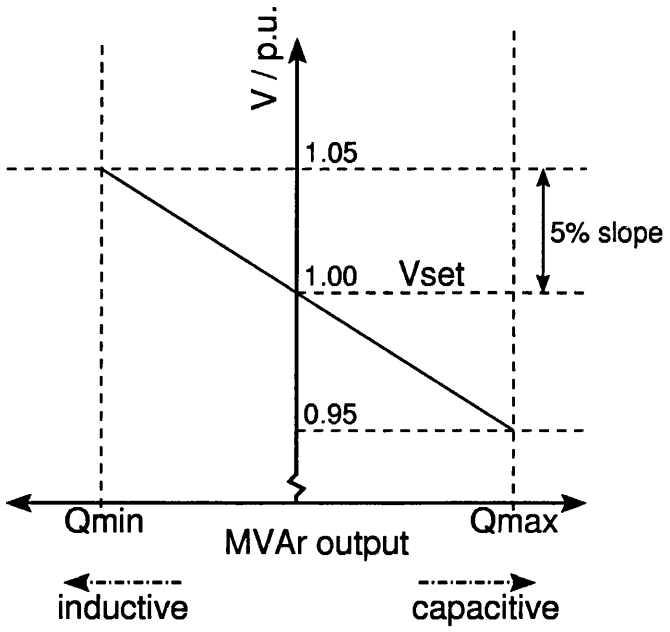

Figure 3.2: Example of a typical voltage-VAr characteristic of a voltage controlled compensation device

This characteristic is described by the voltage set point, which is the observed voltage that causes zero net VAr output, and the characteristic's slope, which is the relationship between the deviation in voltage from the set point and the VAr output of the device. These values for the slope and set point, which remain constant during contingent states, are also optimised by SCORPION.

3.5.4 Conclusions

Optimisation algorithms

A number of optimisation algorithms have been used to solve the RCP problem:

- linear programming
- integer programming
- mixed linear and integer programming

- tabu search
- heuristic
- simulated annealing
- genetic algorithms

As will be discussed in greater detail in section 4.1.2, there are a number of features that set GAs apart from other optimisation techniques:

- GAs perform a parallel search. This makes them more tolerant of local minima. Due to the fact that the RCP problem space is multi-modal such a feature is very important to successful optimisation.
- GAs use payoff information (fitness or objective functions) directly for the search direction, not derivatives or other auxiliary knowledge. This enables GAs to deal with the non-differentiable, discontinuous and non--smooth GCP problem space, without resorting to the approximations that are characteristic of other approaches.
- GAs use probabilistic transition rules, not deterministic rules, to select the next generation. This enables them to search complicated and uncertain areas to find the global optimum, thus making them more flexible and robust than conventional methods.
- GAs are naturally integer based methods. As highlighted by Aoki *et al* [1988] and Chung and Leung [1999], the RCP problem does not involve any discrete parameters, and some variables, such as the available sizes of compensation devices, are highly discontinuous.

These features make GAs robust, parallel algorithms that adaptively search for the global optimal point [Ma and Lai, 1997], and have been found to produce excellent results when applied to the RCP problem, yet their application to

RCP problems of a realistic magnitude has not been fully exploited. This work investigates the application of GAs to realistic RCP problems.

Contingencies

As unfortunate as contingencies are, they are inevitable and must be considered in the RCP problem. In fact, due to the voltage and power flow disturbances that accompany contingencies, adequate reactive power compensation is even more important [Thomas *et al*, 1995; MacQueen *et al*, 1998]. Also, as has been argued , due to the relatively short times such contingent states last, the operational cost of contingent systems is secondary to their security and only the operational cost of the intact system has a significant effect on long term cost.

When contingencies have been supported, the various system states have been optimised separately, after which another stage is required to formulate a unified solution. In this work, by optimising for all system states simultaneously, an improved final solution may be found.

Load flow and sub-problems

As discovered by Bridenbaugh *et al* [1992], the use of Optimal Power Flow algorithms is required to fully evaluate and validate solutions to the RCP problem, and as Lee *et al* [1986] demonstrated, OPF can be successfully moved into a sub-problem of the planning exercise. Vaahedi *et al* [1999] pointed out that existing tried and tested OPF tools can be successfully used by RCP implementations.

Chapter Four

Genetic Algorithms

This chapter covers the optimisation techniques referred to as *Genetic Algorithms* (GAs). GAs are search algorithms based on the mechanics of natural selection and natural genetics.

The term *Natural Selection* was coined by Charles Darwin to describe his theories about the mechanism for evolution and diversity: "the preservation of favourable variations and the rejection of injurious variations [Darwin, 1859]". Often referred to as *survival of the fittest* – slightly misleadingly: death of the weakest would probably be a better catch-phrase – Darwin describes how the variation between organisms in a competitive environment suggests that some of these organisms will have beneficial variations and will therefore have an advantage: a greater chance of survival. Successful organisms will tend to proliferate and so assume a greater share of the population. This leads to the conclusion that successful variations themselves will, likewise, tend to persist and through reproduction proliferate.

The science of genetics started when it was realised that an organism does not pass on a copy of itself to the next generation, but instead provides it with the genetic material containing the information needed to construct a progeny organism [Lewin, 1997]. DNA is that genetic material. When two organisms come together in sexual reproduction, the offspring will be described by a combination and reformation of the DNA from both parents: a unique individual that expresses characteristics taken from both parents.

The basic implementation of a GA, introduced by Holland [1975] and further developed by Goldberg [1989] is called the *Simple Genetic Algorithm* (SGA). It models these concepts by performing the search using a *set of solutions* to

a problem: an analogy to a *population* of competing organisms. Instead of these organisms living in an environment, they exist as points in the problem space: candidate solutions to the problem. A solution's ability to solve the problem is analogous to an organism's *fitness* to survive in its environment. The optimisation is performed as an iterative process: at each iteration (or *generation*) the population is processed with a number of *genetic operators*: crossover, mutation, and reproduction, respectively. These model the effects of reproduction and "survival of the fittest" found in nature.

4.1 Mechanics of the Simple Genetic Algorithm (SGA)

The SGA is the first, and most simple, of the GA type techniques, but still serves as the basis for more sophisticated techniques, which are generally an SGA with one or more operators modified.

The SGA search uses a *population* of binary strings, each representing a point in the problem space: a possible solution to the problem. The simplest coding formulation is achieved with binary concatenated strings. In this formulation, each variable of a candidate solution is encoded within its own sub-string, the complete string being the concatenation of all the substrings. Section 4.2 discusses string coding in more detail. Note that the initial population is generally produced by generating random strings.

The measure of a solution's ability to solve the problem is called its *fitness*. Each string is evaluated by some sort of *fitness function*, which assigns to each string a value representing its fitness to solve the problem. The fitness function represents the problem itself, and exactly how it assigns fitness values to candidate solutions is problem specific and not an issue for the GA itself. In the RCP problem, for example, the fitness function is where each string is

converted into the specification for a planning scheme, load flow is carried out, costs are calculated and so on. The fitness function them evaluates some number which represents the strings ability to optimise the problem: its fitness. Note that in this work, higher fitness values are associated with fitter – that is, better – candidate solutions, as is conventional when discussing GAs. This process is illustrated in figure 4.1, in which each different string is represented by a different colour.

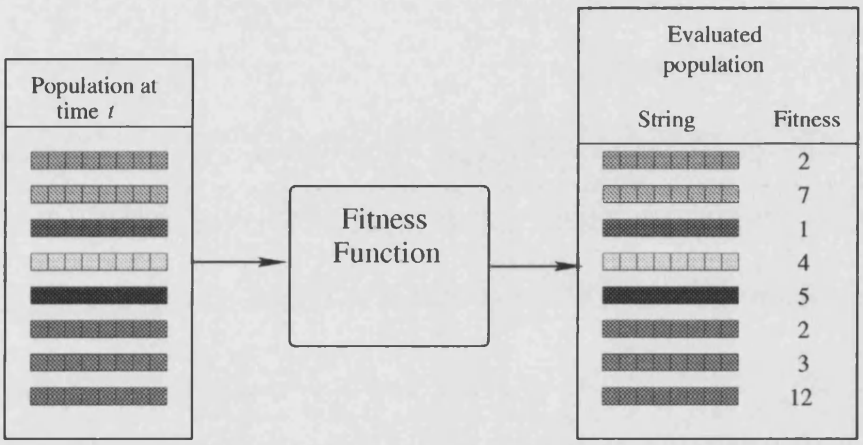

Figure 4.1: Fitness evaluation

The figure illustrates how each string passes through the fitnesses function and is associated with a fitness value. Note that the order of strings within a population is effectively random. At no time is the population actually sorted or ranked. The SGA does not use information relating to the position or index of a string within the population.

Optimisation is performed iteratively across the whole population, in the sense that the algorithm attempts to improve the fitness of the population as a whole, not for each string individually. At each time instant, or *generation*, the population is subjected to a number of *genetic operators*: *crossover*, *reproduction* and *mutation*. These operators attempt to find improved solutions by producing new strings using the strings of the previous generation.

4.1.1 Genetic operators

Reproduction

The reproduction operator represents a *survival of the fittest* mechanism. From the population at time t, a new population is selected. To select this population, a fitness-weighted random algorithm is used: the fitter a particular string, the higher the probability it will be selected. This is performed using a *roulette-wheel* model – this is roulette-wheel in the casino sense. When the roulette-wheel is spun, the string corresponding to the slot into which the ball falls is selected for reproduction; this can be repeated until the new population is acquired. Each string is allocated a number of slots on the roulette-wheel; the fitness of each string, in relation to the total population fitness, determines how big a share of the roulette-wheel it is associated with. This is demonstrated in figure 4.2, which illustrates the reproduction procedure on the population from figure 4.1. The resultant set of strings forms a temporary mating pool on which further operations will act.

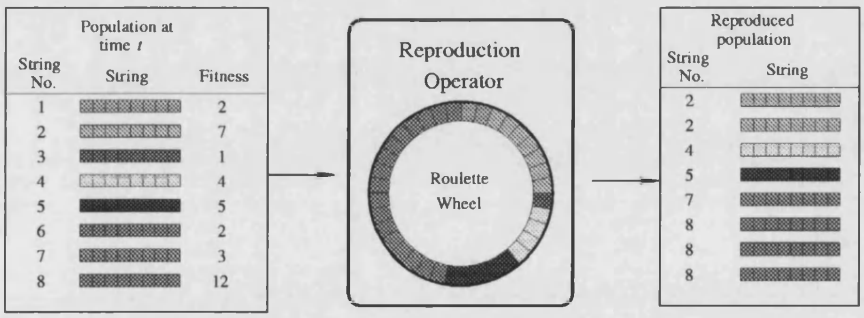

Figure 4.2: The reproduction operator

The total population fitness is 36; as the roulette-wheel is divided up into 36 slots, each string receives a number of slots equal to its fitness. For example, string one is associated with two slots and string eight, twelve slots. To select the new eight-string population, the roulette-wheel is spun eight times. In this example, the post-reproduction population is made up of two copies of string two, one copy of each of strings four, five and seven, and three copies of string

eight; strings one, three and six have no copies in the new population. This is, coincidentally, the statistically most likely resultant population, although it should be stressed that the resultant population could have been made up of any combination of the original strings; the point is that the fitter strings are *more likely* to be in the resultant population.

Crossover

Crossover models the exchange of genetic information that occurs within a mating pool. The strings of a population are randomly paired off and randomly-selected adjacent sections of the strings of each pair are then removed and exchanged between the two. This is represented in figure 4.3 which follows on from the previous two figures. Here, the first randomly selected pair is made up of strings two and eight (which is quite probable due to their frequency in the population); the randomly selected crossover point is between bits three and four in this instance, and this results in two strings made from the first three bits of one string, and the remaining five from the other.

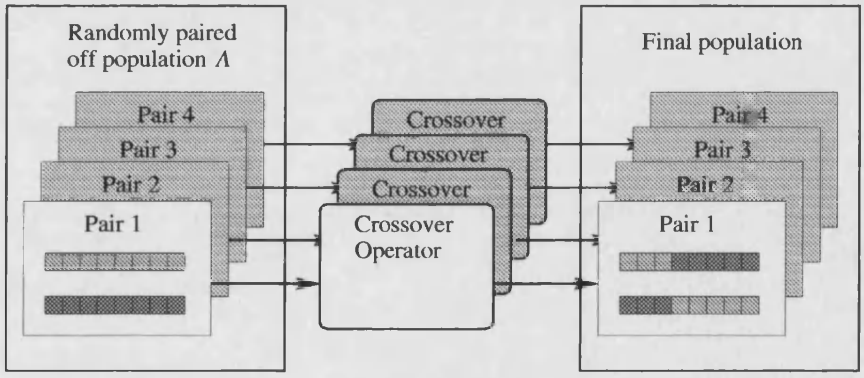

Figure 4.3: The crossover operator

Mutation

The natural effect of errors occurring in genetic code – mutation – is modelled as follows: any bit copying operation is subjected to a probabilistic chance of a copying error. In fact, mutation can be implemented as a separate operation by taking the post-crossover population and selecting bits to be mutated at random, with no loss of functionality. The chance of mutation occurring is small, but has been shown to be essential in the operation of the SGA [Goldberg, 1989], helping it discover new areas for investigation. Finishing our example, figure 4.2 shows what would happen if mutation occurred in the second bit of the string shown: the bit has been mutated from a one into a zero.

Figure 4.4: The mutation operator

Optimal solution

Of course, the whole point in optimisation is to find an optimal solution. Depending on the particular implementation of GA used, there is not necessarily any guarantee that the optimal solution found will still exist in the final solution: it may have been destroyed at some point by, for example, mutation. Therefore, every time the population is evaluated, the members of the population must be examined for the occurrence of new optimal solutions, and, if found, stored for later comparison or presentation. It is also sensible to take actions to maintain this best solution within the population. For example, if it is found that the population no longer contains a previously found, and optimal, solution, replace the weakest member of the current population with the best found so far.

4.1.2 Features of genetic algorithms

The features that set GAs apart from other optimisation techniques are:

- GAs perform a parallel search: as the GA simultaneously optimises a set of independent candidate solutions, not a single solution. The GA is more tolerant to local optima within the search space, as the GA can combine the actions of optimisation and exploration. The set of solutions can, therefore, move over hills and valleys, allowing global optimisation.
- GAs use payoff information (fitness or objective functions) directly for the search direction, not derivatives or other auxiliary knowledge. GAs can therefore deal with non-smooth, discontinuous and non-differentiable functions. This property relieves GAs of approximation assumptions which are often required for other optimisation methods.
- GAs use probabilistic transition rules, not deterministic rules, to select the next generation. This enables them to search complicated and uncertain areas to find the global optimum, thus making them more flexible and robust than conventional methods.
- GAs are naturally integer based methods.

These features make GAs robust and parallel algorithms which adaptively search for the global optimal point [Ma and Lai, 1997].

4.1.3 Directed Evolution

As an interesting example of how powerful and flexible optimisation algorithms based on evolution can be, a process closely analogous to the mechanics of the GA is being used by biochemists to engineer enzymes with specific characteristics. The technique is known as *directed evolution*, for which Arnold

et al [2001] provided an excellent introduction . As Arnold *et al* point out, "the basic algorithm of directed evolution methodology mimics, in some ways, that of natural evolution. The two key steps are generating molecular diversity and identifying the improved variants."

An example of an objective of such a technique would be attempting to improve the thermostability of an enzyme so that it achieves catalytic efficiency at a higher temperature than its *wild-type*: the naturally occurring variant. The directed evolution approach works with an enzyme *library*, which is composed of variants on the wild-type and serves as the *population* (in GA terminology.) The technique attempts to improve the performance of these various enzymes by an iterative methodology. The most widely used approach to generate diversity are *random point mutatagenesis* and *in vitro recombination*. Random mutatagenesis can be conveniently achieved using an error prone version of the *polymerase chain reaction* (PCR): a gene amplification technique. This has the added advantage of a controllable mutation rate. From this point some screening technique would be applied to positively select for beneficial variants – in the above example, this would involve screening for improved thermostability. Finally, a recombination method is used to combine beneficial mutations.

Of particular note is the observation, by Arnold, that a "low error rate (2-3 base substitutions or ≈ 1 amino acid substitution per sequence per generation) accumulates mostly adaptive mutations". See section 4.6.3 for a discussion of the effects of different mutation rates.

4.2 Representation of candidate solutions: string coding

An important concern when implementing a GA is how the candidate solutions are encoded within strings. As mentioned, each string represents a point

in the problem space, but how this string relates to the actual parameters of the problem is important. The simplest idea for string coding is that of binary concatenated strings. This is best described using an example. Assume that the problem at hand involves finding the optimum values of two integer parameters A and B. Parameter A can be any value between zero and four, B between zero and eight. Taking each parameter on its own, this means that a minimum of three binary digits (*bits*) are required to represent the five possibilities for A and four bits for the nine possibilities of B. This is summarised in table 4.1

Table 4.1: Example of formulating a string representation

Parameter	Range	Number of possible values	Bits required
A	0 - 4	5	3
B	0 - 8	9	4

A string would therefore be a seven bit number: the first three digits encoding parameter A and the final four encoding parameter B. Conforming to this encoding scheme a typical string, S, could therefore be $S = 0100101b$ (the b indicates that the number is binary.) This would be split into two substrings, and hence decoded into the two parameters:

$$S = 0100101b \quad \Rightarrow \quad \begin{array}{rcl} A & = & 010b = 2 \\ B & = & 0101b = 5 \end{array}$$

In this manner, any number of parameters can be encoded in a string: each being encoded, using a binary representation, within its own substring.

It is of note that using binary encoded sub-strings does present a problem: strings can encode out-of-range values. Considering the previous example, where parameter A could take any value between zero and four, the three bit substring could represent any value between zero and seven. If the encoded value was found to be greater than four, and hence out of range, the string would need to be declared invalid or the value mapped back into the valid

range.

Declaring strings invalid would not be advisable when considering large real world problems as many strings, especially in the highly random initial populations, may cause such a violation. Invalid strings would need to be removed from the population, and this would result in a drain in the total information contained within the population as a whole: information which could well be useful.

Mapping can also be problematic. Three techniques present themselves: rounding, reflecting and wrapping. A rounding approach simply rounds values down to a legal value, reflection maps values back into the legal range by reflecting values around the maximum legal value and a wrapping approach subtracts the maximum value from the value, in effect, wrapping the value between the limits. These various techniques are illustrated in table 4.2.

Table 4.2: Example of techniques for mapping values onto a 0 to 4 range

Original value	0	1	2	3	4	5	6	7
Rounded value	0	1	2	3	4	4	4	4
Reflected value	0	1	2	3	4	3	2	1
Wrapped value	0	1	2	3	4	0	1	2

Unfortunately, all these methods can be held responsible for the same undesirable effect: problem space distortion. Mapping values by any of the above methods actually causes, to the GA, misleading fitness values. Consider an example: the original problem space for the example problem given above is as illustrated in figure 4.5. The following three figures 4.6, 4.7 and 4.8 show the effect on the problem space of the three mapping approaches. All three can be seen to distort the problem space (from the perspective of the genetic algorithm.) For instance, in the case of reflection mapping a parameter value of seven will appear to be very good and the GA will give such a string more

attention in the search – through reproduction, for example – than perhaps it should.

The important point to note is that, although the GA may still be able to find the optimal solution, generally such distortions will make the problem much more difficult for the GA to optimise, as any relationship between string values and fitness that may exist will be further hidden.

4.3 Problem expression

As discussed in section 4.1, the SGA uses a fitness function to assign numeric values to candidate solutions. These values represent the candidate solution's ability to solve the problem; the GA then acts to maximise the fitness of solutions. This section discusses the formulation of a generalised problem expression. Note that often objective functions, which represent factors that require optimisation, and penalty functions, which relate to penalties for violating limits and so on, are considered separately. This semantic distinction does not change the mathematical formulation of the problem. In this work all measures of a solution's performance are considered to be objective functions: penalty functions are mearly objective functions where the objective is to remove the penalty.

4.3.1 Performance functions

Performance functions are the first level of solution evaluation. Once a candidate solution has been evaluated within the problem domain, two sets of datum are available. The first set contains any data specified or implied by the candidate solution. These values are stored within the *solution-variable vector* ς. Secondly, there is the set of data returned by the evaluation of the solution-variable vector. These values are found within the state-variable vector, σ.

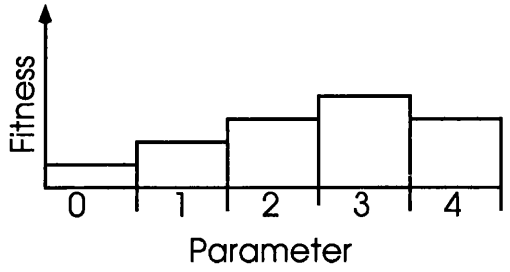

Figure 4.5: Original problem space

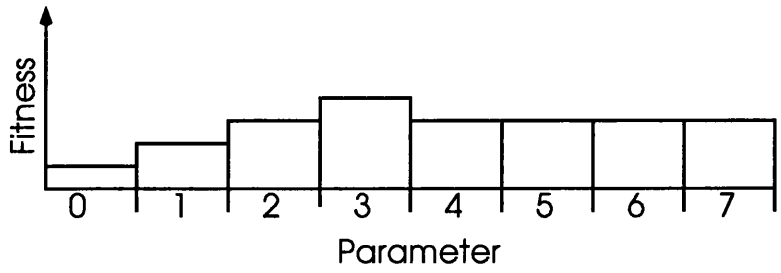

Figure 4.6: Problem space with parameter rounding to a 0 to 4 range

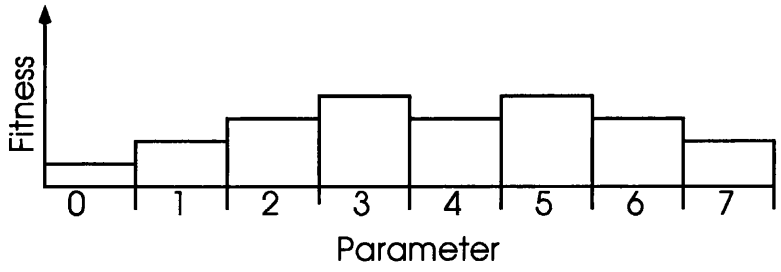

Figure 4.7: Problem space with parameter reflecting to a 0 to 4 range

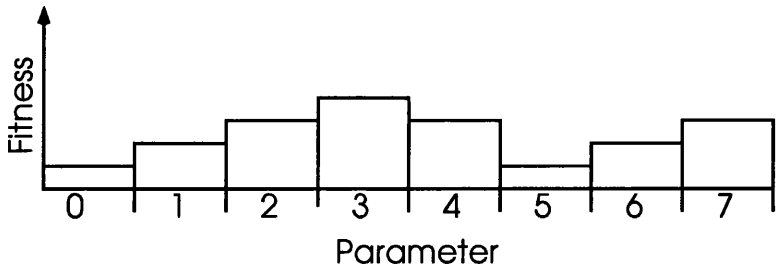

Figure 4.8: Problem space with parameter wrapping to a 0 to 4 range

A performance function will use these raw values to evaluate a numerical value that is proportional to the solution's performance from some perspective: they express one aspect of the problem to be optimised. Typical performance measures include cost, error measurement an so on. Note that at this level of problem expression the functions are not constrained by the algorithm, so the actual formulation of a performance measure is completely arbitrary.

These performance functions are described by:

$$\rho(\varsigma, \sigma) \in \mathcal{P} \tag{4.1}$$

where

$\rho(\varsigma, \sigma)$ = performance function

$\mathcal{P}$ = set of all performance functions

ς = vector of the solutions variables

σ = vector of the state variables

Here, ς expresses all data described by a candidate solution and σ expresses all data available from implementation of that candidate solution. For example, in the compensation planning problem, the vector σ will express results acquired from modelling the power system with the candidate solution implemented upon it. There must be a performance function for each aspect of the problem that must be optimised.

4.3.2 Objective functions

Once the complete set of performance functions have been expressed, the objective functions themselves can be formulated. Objective functions are the layer of abstraction between the measures of performance and the fitness function. The aim of this layer of extraction is to form a minimal set of maximisation functions that mathematically express all aspects of the problem: the set of objective functions, Θ. Each objective function expresses a subset of the performance functions, numerically combined, using a weighted sum approach. Note that the weighting factor, ω_ρ, may be set to zero to exclude the performance function ρ from the particular objective function θ, or set to a negative value to turn a minimisation function into the required maximisation function.

The problem, at this layer of abstraction, can be described by:

$$\max_{\varsigma} \theta(\varsigma, \sigma) = \sum_{\rho \in \mathcal{P}} \rho\omega_\rho \quad : \quad \theta \in \Theta \tag{4.2}$$

where

θ = objective function

Θ = set of all objective functions

ω_ρ = weighting factor of performance function ρ

In natural language, equation 4.2 states that the GA is required to find the value of ς that maximises all objective functions, which are all functions of ς and σ.

4.3.3 Fitness functions

At this point the formulation of the problem becomes GA dependent. As discussed, the SGA uses a fitness function to assign a fitness value to each string. Not all GAs require this assignment. To see why, consider the ways the GA uses fitness values: selection of best solution ever and allocation of each share of roulette wheel for roulette-wheel selection. Selecting the best solution only requires relational information about competing solutions: only a *better than* operator is needed. It is only roulette-wheel selection which requires quantitive information about solution fitness. Roulette-wheel selection is a *fitness-proportionate* selection method. Stochastic Universal Sampling *SUS* is similar to roulette-wheel selection except that a set of individuals are picked simultaneously, based on a random choice of a given number of positions spaced equally around the wheel [Aldridge *et al*, 1999]. Section 4.4 presents some advanced selection methods which are not fitness-proportionate.

To implement fitness-proportionate selection, a single value is required for each candidate solution. This is derived from the objective functions, Θ.

Ideally, only one objective function would be required to express the complete problem, $\mathcal{P}$, but difficultly arises when no correlation exists between subsets of $\mathcal{P}$. For example, a classic example of this is in a factory where a new process is being developed, and optimisation is required. Two performance-measures of a candidate solution are the investment cost and operational costs. These can easily be formulated into maximisation functions and then combined into a single objective function. Problems arise because, in this example, the process is physically dangerous for the people operating the machinery. The method being used to model candidate solutions, whatever it might be, returns a state variable that represents the expected number of injuries as a function of time. Arguably the number of injuries could be converted into a running cost by evaluating the cost of lost work-hours, compensation claims and so on. If the developers do not wish to be so openly mercenary, another formulation needs to be developed. Now there are two non-correlating sets of performance functions: cost functions and *risk* functions, and therefore two objective functions. As this problem requires the simultaneous optimisation of more than one objective function, the problem is said to be a multi-objective optimisation function, and requires special consideration. The simple approach is by using the weighted sum method to combine the two objective functions, which is presented below; more sophisticated techniques are presented in section 4.4.

To evaluate the single fitness value required by fitness-proportionate selection, the values from each objective function can be added together to form an intermediate fitness value $\check{\mathcal{F}}$:

$$\check{\mathcal{F}} = \sum_{\theta \in \Theta} \theta \omega_\theta \tag{4.3}$$

where

$\check{\mathcal{F}}$ = intermediate fitness value

ω_θ = weighting factor of objective function θ

This intermediate fitness value may be unacceptable. Firstly, because most fitness proportionate selection algorithms (roulette-wheel selection, for instance) require non-negative fitness values. Secondly, as will be discussed later in this section, highly fit individuals risk dominating the population by being overly selected. This reduces the diversity of the population and risks losing valuable genetic information. In order to prevent this, intermediate fitness values are typically scaled linearly [Aldridge *et al*, 1999] to evaluate the final fitness value, $\mathcal{F}$:

$$\mathcal{F} = \frac{\check{\mathcal{F}} - \mathcal{F}_{\text{low}}}{\mathcal{F}_{\text{high}} - \mathcal{F}_{\text{low}}} \tag{4.4}$$

where

$\mathcal{F}$ = fitness of candidate solution

$\mathcal{F}_{\text{low}}$ = lowest intermediate fitness value in the population

$\mathcal{F}_{\text{high}}$ = highest intermediate fitness value in the population

Equation 4.4 will assign fitness values between zero and one. Note that as an implementing issue one must account for the situation where all the intermediate fitness values are the same, as this will cause all fitness values to be zero, which can cause roulette-wheel selection to go into an infinite loop.

The problem of using a fitness value quantitatively becomes apparent when the distribution of points is considered. Imagine a problem space with large areas of infeasibility that are greatly penalised through some objective. The fitness of infeasible solutions will, therefore, be very low. Initially, the randomly generated population is full of low scoring solutions, as expected. Then one solution is generated that finds itself in a feasible region. This solution may be so much better than the others that it gets reproduced massively and dominates the following generation. This results in losing most

of the information that the random initialisation injected into the population. Subsequently, there will not be enough information in the population for the crossover operation to generate new candidate solutions, and the risk of converging on a local optima is high.

Assuming that the population avoids this situation and evolves into a set of feasible and varied solutions, there is another pitfall looming. If the mutation operator generates an infeasible solution then the relative difference between the feasible solutions will be massively reduced. That is, from the perspective of the reproduction operator, the feasible solutions – by comparison with the infeasible solution – all look very similar. If infeasible solutions are regularly generated, the convergence of the GA will be adversely affected.

4.4 Advanced selection methods and multiobjective optimisation

Roulette-wheel selection, as implemented by the SGA, is an example of *fitness-proportionate* selection. There are two major drawbacks to this type of selection. Firstly, is simply the fact that they require a single fitness value, and secondly is the fact that this value is used quantitatively, as opposed to the stochastic nature of evolution.

As discussed, if non-correlating objectives exist it becomes very difficult to numerically formulate the fitness value. If a weighted sum approach is to be used, either expertise and heuristics or detailed knowledge of the problem space is required – the range of values expected for each objective, and so on – neither of which may be available. In addition if there is a relatively large separation between "good" and "bad" solutions, there can be problems with scaling. Both these problems can be avoided through advanced selection methods: namely, ranking and tournament [Aldridge *et al*, 1999].

Ranking and tournament selection techniques avoid the problematic use of quantitive fitness values by only using qualitative information: these methods only need a comparison operator that discriminates one solution as being "better than" another. An example of how this comparison can be made, even for multi objective problems, is by searching for *Pareto-optimal* solutions, as defined in section 4.4.3.

4.4.1 Ranking Selection

To perform ranking selection the population is first sorted and ranked, the best solution being placed first, the second best second, and so on. The new population is then formed by selecting solutions according to their rank. The best solution receives the highest share of the next generation, perhaps ten percent of the new population will be made up of copies of this best solution, for example. The solution that ranked second receives a smaller share, eight percent for example, and so on until the new population is filled. Ranking based selection is illustrated in figure 4.9.

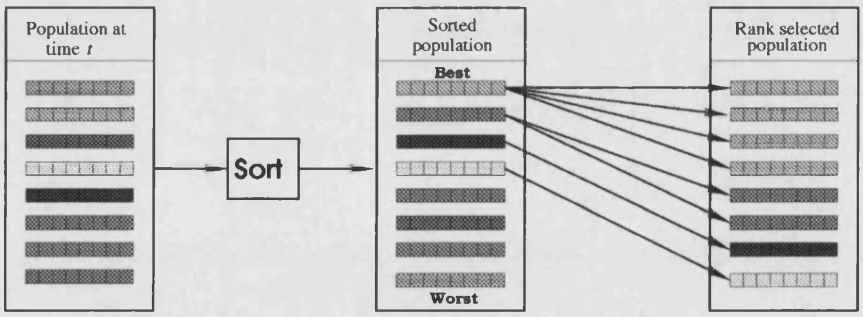

Figure 4.9: The ranking selection method

In the example, the best solution received four copies, the second best, two, and the third and forth best, one copy each. Note that for this scheme the four worst solutions did not, and cannot, receive places in the new population.

4.4.2 Tournament Selection

Tournament selection is illustrated in figure 4.10, in this example the relative performance of the strings is the same as in figure 4.9.

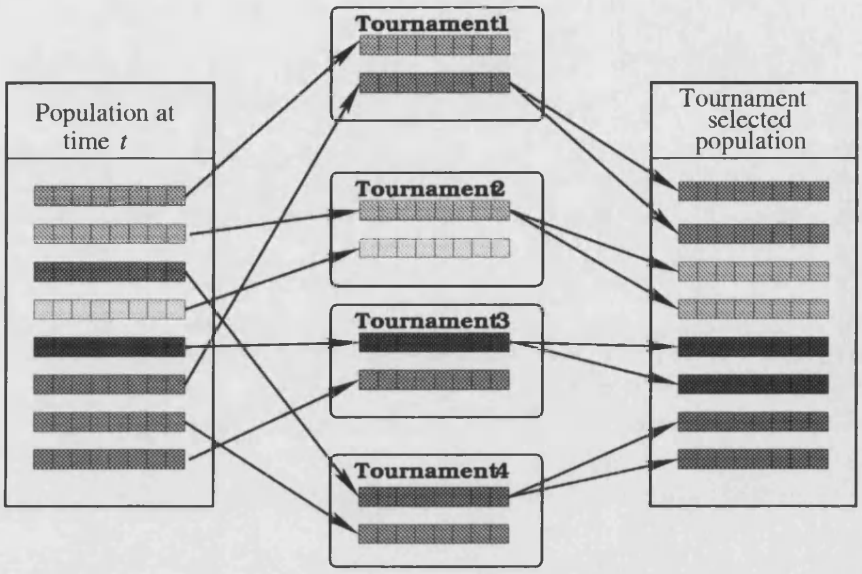

Figure 4.10: The tournament selection method

Tournament selection involves, firstly, dividing the population randomly into a number of smaller groups; in figure 4.10 four groups of two have been randomly chosen. Each group in turn is then examined and the best solution is chosen as the winner. The winner then replaces the loser in the resultant population. The figure shows how the post-selection population is made up of the four winners: each having two instances of themselves. It is interesting to see that in this example the red solution was pitched against the brown in tournament 4. As these are both fairly bad solutions, neither would have ever made it through ranking selection, but here the red solution has won its tournament and been selected. This is an important characteristic of tournament selection: there is a chance that any candidate solution – apart from the very worst – can be selected. This is beneficial to the search ability of the GA, as vital information can be present only in bad solutions, but which could in future populations form part of the optimum solutions, is preserved.

Another point to highlight is that in the example given the best string in the post selection population has the same number of occurrences as the worst. This could look like a weakness in the algorithm, but it must be remembered that the process of selection is repeated every iteration, and weak solutions will eventually be removed as the population concentrates on the higher scoring areas of the problem space.

4.4.3 Pareto-optimality

An established method for defining the characteristics of optimal solutions for multi-objective problems is Pareto-optimality. At the heart of this technique is the concept that a solution x is "better" than a solution y if x *dominates* y. In natural language, x *dominates* y if x is better than y for at least one objective function, and is no worse for any of the others. A solution is Pareto-optimal if it is not dominated by any other solution.

Let there be a number of objective functions, $f_1, \ldots, f_z$, each mapping from the problem space S to the solution space R, that is $\forall i \in \{1, \ldots, z\}, f_i : S \rightarrow R$. The optimal solution, $x \in S$, maximises every $f_i(x)$. In non-trivial problems, it will not be possible to find a solution to a multiobjective optimisation problem. Successfully maximising one of the component functions will typically degrade performance in another. In practice, solutions are required such that no one component function can be improved without sacrificing another. Such solutions are called Pareto-optimal; they are non-dominated solutions. Formally, of two solutions $x, y \in S$, x dominates y if $\exists i \in \{1, \ldots, z\}$ such that $f_i(x) > f_i(y)$ and that $\forall j \in \{1, \ldots, z\}, f_j(x) \geq g_j(y)$. An interesting characteristic of this algorithm is that if two solutions are both individually better in one way or another – so for x and y it may be found that $\exists i \in \{1, \ldots, z\}$ such that $f_i(x) > f_i(y)$, and, $\exists j \in \{1, \ldots, z\}$ such that $f_j(y) > f_j(x)$– neither solution is dominant and the solutions are – in the Pareto sense – equal.

When the GA has finished optimising there may be a set of solutions, none of which are dominated by any other. A multiobjective GA therefore presents a set of optimal solutions, and it is up to someone, or something else, to decide which solution to adopt as strategy. Thinking back to the "dangerous factory" example, where a multiobjective formulation was developed to avoid the problem of having to financially evaluate employee welfare, it is a fact of life that such a decision still needs to be made eventually.

Figure 4.11 shows an example of the final population of a Pareto-optimisation run. Here, two objective functions are being maximised: objective A and objective B. In the figure, the non-dominated solutions are marked by a circle and the dominated solutions by a cross. Notice that any candidate solution to the left and below another solution is dominated by that solution and therefore any solution in the shaded area is dominated. In the case of solution G and B, they are equal in respect to objective B, but solution B is better than solution G in respect to objective A: therefore solution B dominates solution G. In the example, three solutions, A, B and C, are non-dominated and hence Pareto-optimal within the population.

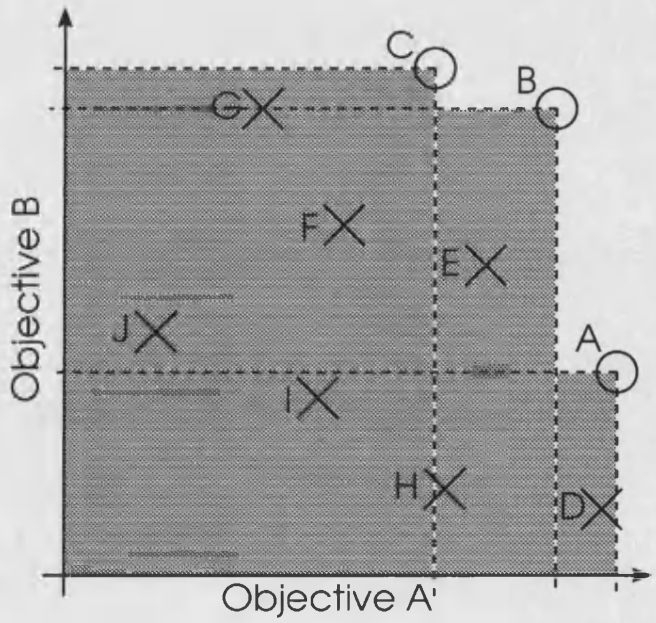

Figure 4.11: Example of Pareto-optimality

4.5 Infeasible solutions

An issue for any optimisation algorithm is what to do when there is the possibility of infeasible solutions, that is, when a solution may be presented for evaluation and all that can be said is that it "doesn't work". If large areas of the search space are infeasible and no measure of performance is available, this can cause significant problems. Until a feasible solution can be found, the GA will just randomly search around the space. When a feasible solution is found the population will quickly converge on this one solution. The worst case scenario in such matters are *needle in a haystack* problems, where the problem space is mostly flat with just one single optimal peak. Such problems are very difficult for a GA which will in fact perform no better than a random search.

On this same line, another even worse situation can occur. Consider a multiobjective problem where a solution involves spending money to improve a process, and there are regions of infeasibility in respect to the evaluating the performance of this process. There are, therefore, two objectives: improve performance and reduce cost. A situation could occur where no feasible solutions are found, which means that the process's performance cannot be numerated and so it must be set at some arbitrary value. The only remaining search pressure for the GA is the solution cost. As the GA attempts to minimise this cost, its search could easily be moving *away* from the feasible regions, which would be disastrous. This highlights an important area for concern when implementing the GA: evaluating a *Measure Of Failure* (MOF). Even if no rigorous formulation can be found for a MOF, any information is better than nothing, and an approximation or rule of thumb may be available. In addition to a MOF, other techniques may be required, for instance some form of heuristic objective switching. For example, if infeasibility is detected, it may be better to forget about the costing of solutions, and to allow the GA to seek out feasible regions without experiencing cost related pressures.

4.6 Genetic algorithm parameters

There are a number of configurable parameters that can affect the performance of a GA.

4.6.1 Population size and the number of generations

The population size and the number of generation are related as they both set the *dimensions* of the search. For example, a population size of ten over one hundred generations, compared to a population size of one hundred over ten generations both require the same number of evaluations of the objective functions. This is significant as for many non-trival problems it is this evaluation that takes the majority of the real time taken up by the optimisation. These two examples given can be seen to be a thin-and-long and a wide-and-short approach. By varying these dimensions, a change in performance may be observed with no change in real time taken. That said, it is uncommon to force such a rigid limit on the number of generations; it is far more common to implement – as for so many other iterative optimisation algorithms – some sort of stopping criteria: for example, after no improvement has been seen for a specified amount of time or some measure of population convergence is observed of exceeded.

Increasing the population size means that a greater proportion of the search space will be sampled, each generation. There is, therefore, a higher chance that productive areas will be discovered by the search: improved exploration. A bigger population also, unfortunately, implies that the chance of any two particular members of the population coming together in crossover is reduced. Once a potential area for exploitation has been discovered by a solution, is may take several generation before enough similar solutions have been generated for such solutions to be crossed over with each other, and hence exploit the area efficiently.

If a smaller population size is employed the relative chance of any two particlar strings exchanging information is increased. It will be more efficient at exploiting a productive solution. Again, the opposing characteristic of a smaller population is that it will contain less entropy, and, hence, less potentially useful information; a small population size can deteriorate the GAs ability to explore the search space efficiently.

Another consideration is the real time taken by the search. Although the number of generations taken to find an optimal solution may seem comparable between two different GAs, the bottleneck of GA optimisation is generally the time required to evaluate a solution. This certainly true in the case of the RCP problem, where it can take up to one second to evaluate a single solution, on a standard computer of the day.

The concept behind choosing a population size is, therefore, to find a population that is small enough to provide good convergence and low real-time taken, and big enough to provide vital exploration.

4.6.2 Crossover rate and reproduction rate

Although this chapter has assumed that during each generation every member is subjected to crossover and reproduction this does not need to be the case. Indeed, it has been found that only selecting a proportion of the population to be involved with such operations can lead to an improvement of the performance of the algorithm [Mantovani *et al*, 2001]. The arguments for such approaches revolve around the concept of controlling the amount of entropy in the population. A high reproduction rate may result in premature convergence on super-fit solutions, loosing valuable information contained in the remander of the population. Crossover, like mutation, is a destructive operator – the parents do not exist in the post-reproduction population – so a high crossover

rate may result in good solutions be destroyed before reproduction has had a chance to copy them.

4.6.3 Mutation rate

The choice of mutation rate is important. Mutation is one of the primary sources for the creation of new information within the population; this provides the GA with the ability to explore the problem space in search of new areas for investigation. If the mutation rate is too low, the GA may prematurely converge on a local minima, having never had the opportunity to discover the global optima. Unfortunately, mutation can also interfere with the other optimisation mechanism of a GA search: exploitation. If the mutation rate is too high, the population can become scattered across the search space, and will be unable to fully converge on the global optima.

The mutation rate must, therefore, be carefully chosen to allow both exploration and exploitation to take place. Techniques have been implemented which use an adaptive mutation rate. Lai and Ma [1998], for example, implemented an algorithm which monitors the convergence of the population. Initially, when the GA is still exploring the problem space, a high mutation rate is used to prevent premature convergence. After a period of time, when the population is observed to be converging, the mutation rate is reduced. It is interesting to note the resemblance between this idea and that of simulated annealing (as discussed in section 3.2.1).

Generally, mutation rates are quoted in terms of, for example, one mutation every few hundred copying operations. This method of expressing mutation rates can, however, be misleading. A current school of thought suggests that meaningful mutation rates should be expressed in terms of the string length itself. A mutation rate of, say, one mutation per three hundred copies becomes far more meaningful when one considers the string length. A string length

of three hundred – which implies three hundred copies required to copy the whole string – compared to a string length of one hundred and fifty, makes the difference between (on average) one mutation per string copy and two mutations. This is far more significant. It has been suggested that an average mutation rate of one per string per copy yields a good compromise between exploration and exploitation. This makes it particularly interesting to note the observation by Arnold *et al* [2001] that when performing directed evolution (see section 4.1.3) a "low error rate (2-3 base substitutions or $\approx$ 1 amino acid substitution per sequence per generation) accumulates mostly adaptive mutations."

4.7 Conclusion

Genetic algorithms are a powerful optimisation techniques that has been successfully applied to many power system based problems – as discussed in section 3.2.1. This work uses a GA based approach to optimise the reactive power compensation placement problem. The following chapter discusses the specifics of the problem at hand, and chapter 6 discusses how the GA was implemented.

In addition to experiments using an SGA, this work has seen the development of a novel GA based on ideas concerning the role of proteins in organisms. This protein coded GA is developed in chapter 7

Chapter Five

Problem definition

The specification for the reactive power compensation planning problem considered in this work is discussed in this chaper.

In this work, the complete set of system states and any supplementary data (installation limits and so on) required by the siting algorithm is referred to as a *study*. Each system state is a *case*, and the intact or primary case is the *base case*. The remainder of the cases are contingencies.

As outlined in section 3.1 the objective of the RCP problem is to *define the specification for the installation of new reactive sources which achieves the maximum benefit at the least possible cost, while satisfying system and operational constraints.* The siting problem is therefore to find that specification. The problem can be broken down into four sections.

- **Formulation of candidate solutions.** Fundamentally, a candidate solution is a specification for the installation of compensation plant that can be physically implemented on the transmission system. As discussed in chapter 3, the variable transformer taps settings and voltage set points setting for existing dynamic compensation are also required to evaluate the solution. To look at it from a different perspective, to evaluate a VAr siting scheme the question has to be asked: if this scheme were to be implemented, how good would it be? To this end, not only do the new reactive sources need to be installed, but, in addition, the system needs to be configured to attain maximum performance from these new sources. So, in this respect a candidate solution is a scheme for VAr compensation expansion plus a set of system configuration parameters.

Section 5.1 describes the generalised form for the expression of the integer parameters of candidate solutions; section 5.2 deals with the parameters for expressing the VAr expansion and configuration; and section 5.3 deals with tap-changing transformers.

- **Implementation of candidate solutions.** Once a candidate solution has been formulated, it needs to be implemented in such a way that its performance can be evaluated. Since candidate solutions cannot actually be physically implemented, some form of simulation or modelling must be carried out. In the case of the siting problem this is generally achieved through steady-state load flow analysis or dynamic power system simulation. In this work, steady-state load flow will be used to evaluate the performance of each case. Section 5.4 discusses how contingencies will be supported. Section 5.5 discusses the use of load-flow in solution evaluation.

- **Evaluation of candidate solutions.** After a candidate solution has been implemented, a wealth of data will exist: load flow data for all cases considered and the details of the installation encoded by the solution. From these data, *measures of performance* of the solution must be evaluated. Here, *measures of performance* simply mean one or more numbers that relate to how good the solution is. The three performance evaluation measures used in this work are voltage deviation (section 5.6.1), reactive power cost (section 5.6.2) and a non convergence measure (5.6.3).

- **Optimisation of candidate solutions.** The crux of the siting problem lies in finding the optimal siting plan. This work investigates the use of a GA for optimisation.

5.1 Stepped limited variables

In the following sections, the variables of the siting plan have a number of traits in common: they are discrete values, each of which has a fixed increment, or step size, between the successive values that they can take, and they are also constrained between maximum and minimum limits. As an example, consider the value of the transformer tap connected to line l: T_l. Note that there are L lines in total.

$$T_l^{\min} \leq T_l \leq T_l^{\max} \quad : \quad l = 1, \ldots, L \tag{5.1}$$

$$T_l = T_l^{\min} + T_l^x \times T_l^{\text{step}} \quad : \quad T_l^x \in \mathcal{N} \tag{5.2}$$

The tap setting must be between the minimum and maximum values specified for that particular tap – that is, between $T_l^{\min}$ and $T_l^{\max}$ respectively – as specified by equation 5.1. The smallest change that can occur in the value of the tap setting, due to physical constraints of the device, is T_l^{step}; this is known as the *step size* of the tap setting. The actual tap setting must be an integer multiple of step sizes above the minimum possible setting $T_l^{\min}$ as described in equation 5.2. Note that this means that a tap setting can be described by this integer-multiple variable: $T_l^x : T_l^x \in \mathcal{N}$, where $\mathcal{N}$ is the set of natural numbers including zero. Many of the variables discussed in this chapter will be subject to equations of the same form as 5.1 and 5.2, and will be referred to as *stepped limited variables*.

Note that for any of the following step-limited values, the minimum and maximum values for a particular variable can both be set to zero. This effectively prevents the variable taking effect at that location: for a line tap limits of zero mean there is no tap on that line and for a busbar expansion limits of zero mean that no compensation can be installed at that location.

5.2 Compensation placement

In this work, it is assumed that only one type of device may be installed at any particular location, but more than one device of this type may be installed. There are three categories of information required to define the parameters of the installation at a particular location. Firstly, the type of device, secondly, the size, and finally any characteristics or operational parameters of that device.

5.2.1 SVCs

For an SVC, the most complex device considered here, the parameters are the inductive and capacitive ranges of the device (L and C) and the set-point of the control characteristic, V, as discussed in section 2.3.2.

Table 5.1 gives the available sizes of the commercial SVCs used by National Grid [National Grid, 2001].

Table 5.1: Available sizes of commercial reactive power compensation devices (National Grid, 2001)

System voltage (in kVs)	Type	Size of single device (in MVAr)	
		Inductive	Capacitive
275	SVC	106	150
400	SVC	75	150

The sizes are for single devices, and more than one device may be placed at a single location as long as this does not contravene any constraint: in particular, the expansion constraints. Note, with care: the capacitive and inductive sizes are not independent. If a node on a 275 kV circuit requires 300 MVArs of dynamic capacitive compensation, the opposing 212 MVArs of

inductive compensation must also be installed, and charged for. In addition, 60 MVAr fixed capacitors are available for installation.

Due to the fact that the capacitive and reactive sizes of devices are not mutually independent, only one integer number, S_b^x, is required to describe the installation at each busbar: effectively, the number of devices. Formally,

$$0 \leq S_b^x \leq S_b^{\max} \quad : \quad \begin{array}{rcl} b & = & 1,\ldots,B \\ S_b^x & \in & \mathcal{N} \end{array} \tag{5.3}$$

and

$$E_b = \begin{cases} 1 & : \text{ if compensation is to be installed at busbar b} \\ 0 & : \text{ if compensation is not to be installed at busbar b} \end{cases} \tag{5.4}$$

where

S_b^x = the number of units of compensation that may be installed at busbar b

E_b = the installation flag: true if compensation is to be installed

The actual size of the device to install can then be evaluated from

$$C_b = E_b \times S_b^x \times C_b^{\text{step}} \tag{5.5}$$

$$L_b = E_b \times S_b^x \times L_b^{\text{step}} \tag{5.6}$$

Where C_b^{step} and L_b^{step} indicate the size of a single device and are taken from table 5.1 for the appropriate system voltage.

Note that by setting $S_b^{\max}$ to zero installation can be prevented at that busbar, and that E_b is actually used to make it easier for the optimisation algorithm to decide whether or not installation should actually be carried out. Without this "E" flag, the only way for the siting algorithm to prevent installation would be to set the size to zero.

The voltage set point V of each new SVC, as well as those of existing SVCs, is specified as follows:

$$V_b^{\min} \leq V_b \leq V_b^{\max} \quad : \quad b = 1, \ldots, B \tag{5.7}$$

$$V_b = V_b^{\min} + V_b^x \times V_b^{\text{step}} \quad : \quad V_b^x \in \mathcal{N} \tag{5.8}$$

where

V_b = voltage set point of device connected to busbar b

Again, setting $V_b^{\min}$ and $V_b^{\max}$ to zero indicates that no device is installed at the associated node.

5.2.2 Fixed reactance

Fixed compensation, which may be either fixed capacitors or fixed inductors, can be described by the size of the device S.

This is represented within the solution by the number of units to install S_b^x and a flag, t_b, to indicate whether the device is a capacitor or an inductor, for each busbar b.

$$0 \leq S_b^x \leq S_b^{\max} \quad : \quad \begin{array}{rcl} b & = & 1, \ldots, B \\ S_b^x & \in & \mathcal{N} \end{array} \tag{5.9}$$

$$t_b = \begin{cases} 1 & : \text{capacitive compensation} \\ -1 & : \text{inductive compensation} \end{cases} \quad : \quad b = 1, \ldots, B \tag{5.10}$$

and

$$E_b = \begin{cases} 1 & : \text{ if compensation is to be installed at busbar b} \\ 0 & : \text{ if compensation is not to be installed at busbar b} \end{cases} \tag{5.11}$$

where

S_b^x = number of devices to install at busbar b

t_b = type of devices to install at busbar b

E_b = the installation flag: true if compensation is to be installed

Again, by specifying $S_b^{\max} = 0$, installation can be prevented, and E is the installation flag that is available for optimisation. As specified by National Grid [National Grid, 2001], sixty MVAr devices will be available for installation.

These values are then used to enumerate the actual size, in MVArs, of the devices to install:

$$S_b = 60 \times S_b^x \times t_b \times E_b \quad : \quad b = 1, \ldots, B \tag{5.12}$$

Note that the convention "positive VArs are capacitive, negative VArs are inductive" is being honoured.

5.2.3 Support of third generation reactive power compensation devices

Although it has been said that the most sophisticated device considered is the SVC, the model used fits any reactive power compensation device that has a linear control characteristic that controls the reactive power generated by the device in sympathy with the observed system voltage. In this way, operation is analogous to that of synchronous generators (see section 2.3.1) and, partially, to that of third generation devices such as STATCOMs (see section 2.3.3). The reason that the third generation devices are not fully supported is that they have the (limited) ability to exchange real power with the system. Therefore, any sited SVC, as specified by the algorithm developed herein, could be

substituted for a STATCOM type device. This would, in fact, be preferable, due to the advantages of STATCOMs over SVCs, as discussed. Even so, it must be stated that the algorithm developed is not a STATCOM placement technique because, due to the real power exchange capabilities of STATCOMs, the optimal locations for SVCs may be suboptimal locations for STATCOMs. Optimal STATCOM placement is, therefore, still a matter for future research.

5.3 Tap-changing transformers

As tap-changing transformers present a method for controlling the voltage profile – and, hence, the reactive power flows – of a system, as discussed in section 2.4, they must be included in the evaluation of a system. Automatic tap-changing transformers will be referred to as *corrective taps* as they have the ability to change tap settings after contingencies and so act to correct the voltage profile. Other tap-changing transformers will be referred to as *preventive taps* as they retain constant tap settings across the cases considered and, therefore, provide constant voltage support. In the RCP problem, only preventive taps are available for optimisation.

5.3.1 Preventive taps

As preventive taps have no inherent ability to change their particular setting and they remain the same even if the system operating conditions change. This means that they have the same value, T, for all system states considered. These values are subject to equations 5.13 and 5.14.

$$T_l^{\min} \leq T_l \leq T_l^{\max} \quad : \quad l = 1, \ldots, L \tag{5.13}$$

$$T_l = T_l^{\min} + T_l^x \times T_l^{\text{step}} \quad : \quad T_l^x \in \mathcal{N} \tag{5.14}$$

To evaluate a siting plan, each value $T_l : l = 1, \ldots, L$ must be enumerated.

OPFLO and target voltages

Optimal Power Flow 02 (OPFL02) is the load flow software developed and used by National Grid. When using OPFL02 the corrective tap settings are not fixed directly; a target voltage, G, is specified instead:

$$G_b^{\min} \leq G_b \leq G_b^{\max} \quad : \quad b = 1, \ldots, B \tag{5.15}$$

$$G_b = G_b^{\min} + G_b^x \times G_b^{\text{step}} \quad : \quad G_b^x \in \mathcal{N} \tag{5.16}$$

(5.17)

This target voltage is for the high-voltage side of the transformer – in practice, the system-side of the tap, as opposed to the generator-side. The taps are then calculated, by OPFL02, to achieve their target voltages, at the specified busbar, for the *base case*. These tap settings are then used for all contingent cases. The final configuration of preventive taps is, therefore, the same as it would have been had the taps been fixed at these values directly.

5.4 Contingencies

As discussed, an important aspect of compensation siting is the consideration of contingencies. In all siting algorithms within the literature an objective is to optimise the base case, but many different approaches for handling contingencies have been used. In the compensation siting papers where contingencies have been considered, the siting algorithm acts to either ensure that the contingent cases are secure or to optimise the performance of the contingent cases. The argument against optimising the performance of contingent states is that any one contingency is rare and comparatively short

lived; greater benefit can be realised by making the base case as economical as possible.

In certain papers [Lai and Ma, 1997a,b, 1998], each contingency is considered as a separate study, with each case receiving a unique siting plan. Although this is interesting for studying the different VAr requirements of contingent systems, from the perspective of a transmission system planner wishing to invest in new reactive plant, these various siting plans would then need to be combined into a single implementable scheme. Simply combining the schemes and implementing the net results could be suboptimal. For example, the single state optimisation could indicate that contingency A required compensation to be installed at busbar one and contingency B required compensation to be installed at busbat two. These sitings could be combined and compensation installed at both busbars when, in fact, a device could be installed at busbar three that would satisfy the constraints of both systems, but at a lower net cost.

The SCORPION package, used by National Grid, starts by performing separate study optimisation and combining the various siting plans, but then uses a heuristic back-tracking algorithm to reduce the net VAr requirement. This can mitigate the unification problem but, as discussed in section 3.5, solutions could still be sub-optimal.

5.5 Load flow: Steady state power system modelling

Candidate solutions will be combined with the study data to form a solution study which can be processed using load flow software.

Load flow [Weedy, 1987] is the term used to describe techniques for evaluating the steady state solution of a network, subject to the constraints and characteristics of power-system plant and complex power flow. The basic network is

described in terms of the nodes or busbars, and the circuits or lines connecting the nodes. Each busbar will have details of its complex generation – that is, the reactive and real generation – and each circuit will have the details of its electrical characteristics: reactance, resistance, susceptance, tap settings and so on. Various extensions to this can be found in certain load flow implementations, including the specification of equations to relate the reactive power output at a node to the system voltage, and the ability to automatically set transformer taps by specifying the required busbar target voltages. Generally, the system is formulated into a matrix representation and solved by inversion or iterative methods such as Gauss-Seidel or Newton-Raphson. Detailed descriptions regarding the implementation of load flow can be found in Freris and Sasson [1968] and Weedy [1987].

Load flow studies are often performed to investigate the following areas:

- flow of MW and MVAr in the branches of the network;
- busbar voltages;
- effect of rearranging circuits and incorporating new circuits onto the system;
- effect of temporary loss of generation and transmission circuits on system loading;
- effect of injecting in-phase and quadrature boost voltages on system loading;
- optimum system running conditions and load distribution;
- optimum system losses;
- optimum system running costs;
- optimum rating and tap transformers;

- improvement from change of conductor size and system voltage.

Many of these are of the utmost importance to the reactive power compensation planning problem, as discussed in section 3.2.1.

An important consideration is that the load flow may not be able to find a feasible solution at all. This is generally caused by no solution being found that does not violate some system or operational limit. The way in which such violations are handled varies between different load flow implementations. Some implementations simply give up when no legal solution can be found, others allow variables to go out-of-limit and then report this excess.

From the results of the load flow calculations the mismatch at each busbar can be evaluated: the difference between the complex power flowing towards the busbar and the power flowing away which should, ideally, be zero. Due to the simplicity of the load flow model, it is normal to accept slight mismatches that may occur at busbars. There are therefore two categories of mismatch that may be observed: critical and acceptable. Critical mismatches mean that no feasible solution can be found and, in essence, there is insufficient generation of power to satisfy the demand; acceptable operation – the supply, at demand points, of electrical power of the correct voltage and frequency – of the power system cannot be guaranteed.

In addition to the steady state solution provided by load flow, the dynamic responses of the power system can also be modelled using time domain simulation of power systems. In this work, only the steady state characteristics of the power system has been considered.

After load flow analysis of each study there will exist a set of data for each case considered: busbar voltages, complex power flows, complex generation and so on. In this work it is assumed that there are C cases where the first case is the base case and the remaining $C - 1$ cases are the contingencies. The vector of state variables for case, c, of the C cases, is represented by s_c, and the

complete set of such vectors by σ, where

$$\sigma = (s_c) \quad : \quad c = 1, \ldots, C$$

5.6 Performance evaluation indices

The following subsections specify a number of measures that will be used to evaluate the performance of candidate solutions. These use either data taken from the results of the load flow analysis, for example, in evaluation of voltage deviation, or alternatively, data taken from the candidate solution itself: primarily, for evaluation of installation costs.

Mathematically, the following data is available for use in performance evaluation functions, using the previously specified notation:

$$\varsigma = (S_b^x, E_B^x, V_b^x, t_b^x, G_b^x) \quad : \quad b = 1, \ldots, B \tag{5.18}$$

$$\sigma = (s_c) \quad : \quad c = 1, \ldots, C \tag{5.19}$$

where

ς = solution variable vector

σ = state variable vector

In the following sections, a number of measures will be formulated. Performance evaluation functions will be expressed by ρ, and all performance measures will be in terms of ς and σ. Thus, a performance evaluation function which measures *characteristicA* will be expressed by the following:

$$\rho_{characteristicA}(\varsigma, \sigma)$$

Now, these performance evaluation measures must be combined to produce

objective functions that may be directly used by the GA; they must be maximisation functions, in terms of the solution variables:

$$\max_{\varsigma} \theta_{anObjective} = \rho_{characteristicA}(\varsigma, \sigma) + \rho_{characteristicB}(\varsigma, \sigma)$$

5.6.1 Voltage deviation

As maintaining the voltage profile is a major objective of compensation siting, many algorithms use a measure of the deviation in the voltages at busbars from some ideal voltage: normally 1 p.u. which would be, for example, 400 kV for a 400 kV transmission system. The algorithm then uses this measure in the formulation of its objective function.

Chen and Liu [1994], defined the voltage deviation at busbar b as:

$$Vdev_b = \frac{\phi(V_b - V_b^{ideal} - \delta v_b)}{V_b} \tag{5.20}$$

where

$$\phi(x) = \begin{cases} 0 & : \text{ if } x < 0 \\ x & : \text{ otherwise} \end{cases}$$

$Vdev_b$ = voltage deviation measure for busbar b

V_b = voltage magnitude at busbar b

V_b^{ideal} = ideal specific voltage at busbar b : nominally set to 1 p.u.

δv_b = tolerance of maximum voltage deviation for busbar b

When the busbar voltage is at its ideal specific voltage (or, more precisely, within the limits $V_b^{ideal} - \delta v_b \leq V_b \leq V_b^{ideal} + \delta v_b$), $Vdev$ will equal zero. As the voltage moves out of these limits, $Vdev$ will increase.

To extend this formulation to support contingencies, the expression $Vdev_{b,c}$ will be used, where:

$Vdev_{b,c}$ = voltage deviation at busbar b for case c

The voltage deviation objective is therefore to minimise the sum of the voltage deviations of all busbars for all cases or to bring all voltages within acceptable limits. The voltage deviation performance evaluation function is, therefore, as follows:

$$\rho_{Vdev} = \sum_{c=1}^{C} \sum_{b=1}^{B} Vdev_{b,c} \tag{5.21}$$

To get the required maximisation function, this sum can be multiplied by minus one. The voltage deviation objective, θ_{Vdev}, is specified in equation 5.22.

$$\max_{\varsigma} \theta_{Vdev}(\varsigma) = -1 \times \rho_{Vdev} \tag{5.22}$$

5.6.2 Reactive power cost

Within the literature, many common themes can be seen when it comes to the formulation of performance evaluation functions: cost data, for example, is always considered, and there is generally some measure of voltage deviation. One measure, however, is particular to SCORPION, the software used by National Grid for VAr compensation planning. That measure is reactive power cost. As discussed, a fundamental aim of this project has been to develop an algorithm that can handle practical sized problems and, in particular, the problem of VAr siting on the England and Wales transmission system. It has therefore been clear from the outset that the software developed would need to

be validated against SCORPION. To provide an even playing field upon which to rigorously compare the two optimisation techniques, it is a requirement that both packages are working within the same problem space.

Reactive power cost combines a number of performance evaluation functions and provides a single objective measured in units of cost. Arbitrarily, pounds sterling are used, but the actual unit is not particularly important; it is their relative values that are important. Remember that the reactive power cost is only considered for the base case.

The performance evaluation measures associated with reactive power cost, ρ_{RPC}, are expressed in equation 5.23:

$$\rho_{RPC}(\rho_{Util}, \rho_{Inst}, \rho_{Fict},) \tag{5.23}$$

These are described in the following list,

- Existing utilisation: ρ_{Util}

  This measures costable generation of reactive power. Costable generation is that from SVCs and synchronous generators. It is associated with a low cost.

  $$\rho_{Util} = \sum_{b=1}^{B} U_{b,1} \tag{5.24}$$

  where

  ρ_{Util} = existing MVAr utilisation

  $U_{b,1}$ = costable MVAr generation at busbar b for the base case

- Installation: ρ_{Inst}

  This measures the installation of new reactive sources: SVCs, fixed capacitors and fixed inductors. It is associated with a high cost. Note that for installation, there is no concept of "case": installation is made across all cases.

$$\rho_{Inst} = \sum_{b=1}^{B} I_b \tag{5.25}$$

  where

$$\rho_{Inst} = \text{MVAr installation}$$
$$I_b = \text{MVAr installation at busbar b}$$

- Fictitious utilisation: ρ_{Fict}

  This measures the use of reactive power from sources that do not exist for all cases. It is associated with a very high cost.

$$\rho_{Fict} = \sum_{c=1}^{C} \sum_{b=1}^{B} F_{b,c} \tag{5.26}$$

  where

$$\rho_{Fict} = \text{fictitious MVAr utilisation}$$
$$F_{b,c} = \text{fictitious generation at busbar } b \text{ for case } c$$

The fictitious utilisation measure may appear a little curious, but it is used internally by SCORPION as, effectively, a measure of failure. Within SCORPION, any busbar has the ability to generate VArs outside its physical limit (after allowing for any compensation installed at the busbar). This means that a deficiency in VArs at any point in the network can be rectified. Obviously, this would still mean that in reality the system would be infeasible, so a very

high costing is used to encourage VArs to be supplied from existing sources or via the installation of new sources. It is not intended that the final solution uses any fictitious generation.

Typically, a cost of one pound is charged for every MVAr of existing utilisation, one hundred pounds per MVAr for new reactive sources, and utilisation of fictitious MVAr sources is charged at one hundred thousand pounds per MVAr. To convert this objective into the required maximisation problem, the sign of these factors must be inverted. Again, considering the multiple contingencies used, the reactive power cost performance evaluationn function, $\mathcal{P}_{RPC}$, can be expressed by the following function:

$$\rho_{RPC} = 1\rho_{Util} + 100\rho_{Inst} + 10000\rho_{Fict} \tag{5.27}$$

The RPC objective cost is expressed as follows:

$$\max_{\varsigma} \theta_{RPC} = -1 \times \rho_{RPC} \tag{5.28}$$

When reactive power cost is not being considered, installation cost (equation 5.25) can be included as a separate objective function:

$$\max_{\varsigma} \theta_{Inst} = -1\rho_{Inst} \tag{5.29}$$

Observe that if installation is included within its own objective function the scaling (by one hundred) becomes meaningless, and is simply replaced by a unit weighting.

5.6.3 Power flow balance: Mismatch and non convergence

As discussed in section 5.5, the results of the load flow analysis may reveal mismatch between generated and absorbed power. If this mismatch is small the solution can be accepted as feasible, but the mismatch can still be used as a performance evaluation measure to apply pressure away from infeasible regions. As discussed in section 4.5, the ability of a GA to find feasible solutions is hampered when no measure of infeasibility is available: it is vital to formulate some *measure of failure* to allow the GA to find the feasible regions.

In compensation siting, when an infeasible solution is presented there are still a number of values – mismatch being one of them – that can provide vital performance evaluation measures for the GA. The values presented by an non-convergent load flow – due to that non-convergence – do not in any way represent the values that would actually be observed in a real system, but can still be used to steer the search towards convergent solutions. In this work a simple method is proposed to get useful, reliable information out of the non-convergent load flow: the number of limit-violating busbars.

$$\Upsilon_c = \sum_{b=1}^{B} \beta_{b,c} \tag{5.30}$$

$$\beta_{b,c} = \begin{cases} 1 & : \text{ if busbar } b \text{ violates any limit in case } c \\ 0 & : \text{ otherwise} \end{cases} \tag{5.31}$$

where

$\Upsilon_c =$ Limit violation measure for case c

These limits are voltage constraints, mismatch constraints, tap setting constraints, and complex-power generation limits. This objective therefore combines all possible causes of infeasibility into one objective:

$$\max_{\varsigma} \theta_{\Upsilon} = -1 \sum_{c=1}^{C} \Upsilon_c \tag{5.32}$$

Another consideration in the case of non-convergent load flow is the effect that other performance evaluation measures may have on the GA search. For example, if all the solutions in a GA's population are suffering from at least one limit violating busbar, the only difference in performance from one solution to the next will be from the cost of implementing the solution, which is evaluated by examining the string itself. This could be problematic, as the only way to get a convergent system may be to install new reactive sources, but the GA will favour the cheaper solutions: the ones with less new reactive sources. For this reason it may also be necessary to use a finer grained measure of infeasibility, for example the voltage deviation or VAr mismatch.

5.7 Conclusion

The reactive power compensation planning problem, as addressed in this work, can now be specified:

Find an solution ς, where

$$\varsigma = (S_b^x, E_b^x, V_b^x, t_b^x, G_b^x) \quad : \quad b = 1, \ldots, B \qquad \ldots \text{sections 5.2 and 5.3}$$

that satisfies the expansion and plant settings limits, as discussed in the relevant sections, and is optimal in terms of one or more of the following objectives:

minimise voltage deviation:

$$\max_{\varsigma} \quad \theta_{Vdev}(\varsigma, \sigma) \qquad \ldots \text{section 5.6.1, equation 5.22}$$

minimise reactive power cost:

$$\max_{\varsigma} \quad \theta_{RPC}(\varsigma, \sigma) \qquad \ldots \text{section 5.6.2, equation 5.28}$$

minimise installation:

$$\max_{\varsigma} \quad \theta_{Inst}(\varsigma, \sigma) \qquad \ldots \text{section 5.6.2, equation 5.29}$$

minimise busbar limit violations:

$$\max_{\varsigma} \quad \theta_{\Upsilon}(\varsigma, \sigma) \qquad \ldots \text{section 5.6.3, equation 5.32}$$

where

$\sigma =$ state variable vectors for the load flows from each case $1, \ldots, C$ of the study incorporating the solution vector ς

As a solution is valid (in terms of system security) only if the number of busbars that violate a system or operational limit is zero, the "minimise busbar violation measure" is essential, and must always be included in optimisation.

Chapter 6 discusses the design and implementation of a solution to this problem.

Chapter Six

Implementation: The Genetic Compensation Placement package

This chapter discusses the implementation of *Genetic Compensation Planning* (GCP): the software developed as part of this project to evaluate the use of GAs for finding optimal solutions to the reactive power compensation planning problem.

6.1 Design considerations

There are a number of design considerations that need to be discussed before details of the implementation can be defined.

Although this work was primarily an academic investigation into the uses of GAs for reactive power compensation, it was also an investigation into how the developed technique could be used to solve real world problems. This added dimension escalated the size and complexity of the problem and, hence, the solution. It is the opinion of the author that, without the design standards and design practices used this solution would never have been realised. Producing a useful and robust solution has had as much to with good design as it has with improving the mechanics of the GA: for example, by developing new genetic operators. This section discusses two of the most influential practices used: object oriented programming and XML.

6.1.1 Object oriented design

Object Oriented Programming (OOP) is a design paradigm in which the structure of the software is a collection of *objects* which represent extractions of real world objects – a car, an engine, a road, and so on – which interact in the same way as they do in the real world; a car *has* an engine and it *uses* a road. Most problems may not permit such clear abstractions as these and it is, therefore, the task of the designer to interpret the problem and extract useful, meaningful and suitable abstractions.

The motivations for OOP are strong. As Handschin *et al* [1998] noted "the object oriented approach has gained widespread importance and acceptance in software development due to the advantages it offers concerning flexibility, expandability, maintainability and data integrity." The application of OOP to problems of this type is not new, an approach to capacitor placement using an object oriented design has been previously demonstrated by Abdullah *et al* [1997].

GCP was implemented using an object oriented paradigm. In addition to bringing all the benefits mentioned above to the implementation of the package, the concepts of OOP were invaluable to the design process. Note that in this work the term *package* is used to describe a collection of objects, or *classes*, that are all dedicated to a particular task. These classes express the chosen abstractions, but note that these are not actually the same as *objects*. The classes – which are written by the author of software – effectively instruct the operating system on how to *make* objects of that class; the objects themselves only existing transiently in the memory of the computer. In OOP terminology, an object of a certain class is referred to as *instant* of that class.

Section 6.3, which discusses the design of the GCP, makes reference to the OOP nature of the design. This is not only to facilitate explanations of how the various techniques operate within what has developed into a large scale

software suit, but also to demonstrate the structure of a real example of a large object oriented design, and illustrate some of the benefits such a paradigm can bring.

Also, the OOP structure is expressed in *Unified Modelling Language* (UML) [Object Management Group, 2001], an industry-standard language for specifying, visualising, constructing, and documenting the artifacts of software systems.

6.1.2 eXtensible Markup Language

There were a number of design standards that were used in the writing of GCP. These were not only used to aid the development of the package, but also to aid any extension or modification carried out in the future by parties unknown.

One of these standards was the *eXtensible Markup Language* (XML). XML is a data format for structured document interchange. XML resembles *Hyper-Text Markup Language* (HTML): in actual fact, HTML is a subset of XML.

The advantages of using XML in software are summarised as follows:

- Standardisation. XML has been fully specified and there is only one specification: if two developers use XML they can be confident they are both using the same thing. The current XML specification can be found at http://www.w3.org/XML/ [W3.org, 2001].
- Computer readable. Although human readable, XML is primarily intended to be computer readable.
- Extensibility. Due to its inherent flexibility, XML can be used for exchanging any sort of data between the applications that implement it. The only requirement is, of course, that both applications are expecting the *same sort* of data.

- Third party support. One of the most appealing aspects of XML is the shear volume of third party support. For example, in GCP a third party function library called *libxml* was used. This library handled the writing and reading of data files, obfuscating the costly requirement for writing application-specific file access routines. Also, there are a number of graphical editors, and so on, for XML data which can serve as part of the interface to GCP. Finally, because of the growing popularity of XML, these third party applications can only become more functional, robust and numerous.

The benefits XML brought to this project cannot be over emphasised. GCP uses XML for all file handling, using the aforementioned library.

6.1.3 Conformance with the National Grid RPC problem

An important consideration during this work has been conformance with the practices and characteristics of compensation planning as carried out by National Grid for the England and Wales transmission system. A primary objective of this work has been to assess whether GAs perform competitively when applied to large planning problems – large in the sense of the size of the transmission system, number of contingencies, number of constrains, number of variable taps, number of busbars that are candidates for compensation, and so on. National Grid agreed to supply realistic test data so that the performance of the algorithm presented in this work could be directly compared to the performance of a productionised compensation siting program: SCORPION– see section 3.5. It is therefore important that the software is compliant with the standards and nuances of the siting problem as specified by National Grid, while not loosing generality.

To be compliant with SCORPION the planning algorithm needs to exhibit the following characteristics:

- the algorithm must act to find a solution that is optimal for the base case and secure for the contingent cases – see section 3.5.1;
- the voltage set-point of the control characteristics of any existing SVCs, in addition to those of newly installed SVCs, must be optimised – see section 5.2;
- the target voltage of the HV side of variable transformer taps must be optimised – see section 5.3;
- the algorithm must act to minimise the reactive power cost – see section 5.6.2;

6.2 Solution Coding

An important design consideration when implementing a GA is the encoding of candidate solutions within strings [Goldberg, 1989]. Initially, in this work a simple binary concatenated string representation was used, as described in section 4.2. Each parameter, as described in chapter 5, will therefore be encoded separately into its own sub-string; the string is made by concatenating each sub-string. The parameters can broken down in the following categories: installation and configuration.

Installation parameters describe the new sources of reactive power to be installed. Each candidate busbar will be associated with a particular sub-string which will specify the reactive sources to be installed. For busbars that may have an SVC installed this sub-string will indicate the size parameter of the device, S_b^x, the voltage set point of the characteristic, V_b^x, and the installation flag E_b. The actual size of the devices can then be enumerated from equations 5.5 and 5.6, and the voltage set points from 5.8.

For fixed-source candidate-busbars the sub-strings will indicate the size, S_b^x, the type, t_b, and the installation flag, E_b. The actually size of the device to

install can then be enumerated from equation 5.12.

The rest of the string is taken up with configuration parameters for the relevant busbars. To recapitulate, these are the variable transformer tap target voltages, G_b^x, and the voltage set points, V_b^x, of the control characteristics of any dynamic compensation that already existed on the system. The actual values may then be enumerated from equations 5.16 and 5.8 respectively. The complete coding is represented in figure 6.1.

To use this coding, the candidate busbars are first grouped and sorted in the following order.

- SVC installation. This group consists of the busbars that are candidates for the installation of a new SVCs and busbars that have existing SVCs that are candidates for expansion.
- Fixed compensation installation. This group consists of busbars that are candidates for the installation of new fixed compensation sources and busbars that have existing fixed compensation that are candidates for expansion. Note that a side effect of this algorithm is that installation incurs costs regardless of the net resultant output. For example, a busbar that was already the site of a one hundred MVAr fixed inductor could have a one hundred MVAr capacitor additionally sited. This capacitor would then be considered an installation cost, but the net busbar VAr output would be zero VArs.
- SVC configuration. This group consists of busbars that have existing SVCs that require that their voltage set points be optimised.
- Preventive tap configuration. This group consist of the busbars that are connected to the high-voltage side of preventive transformer taps that require optimisation.

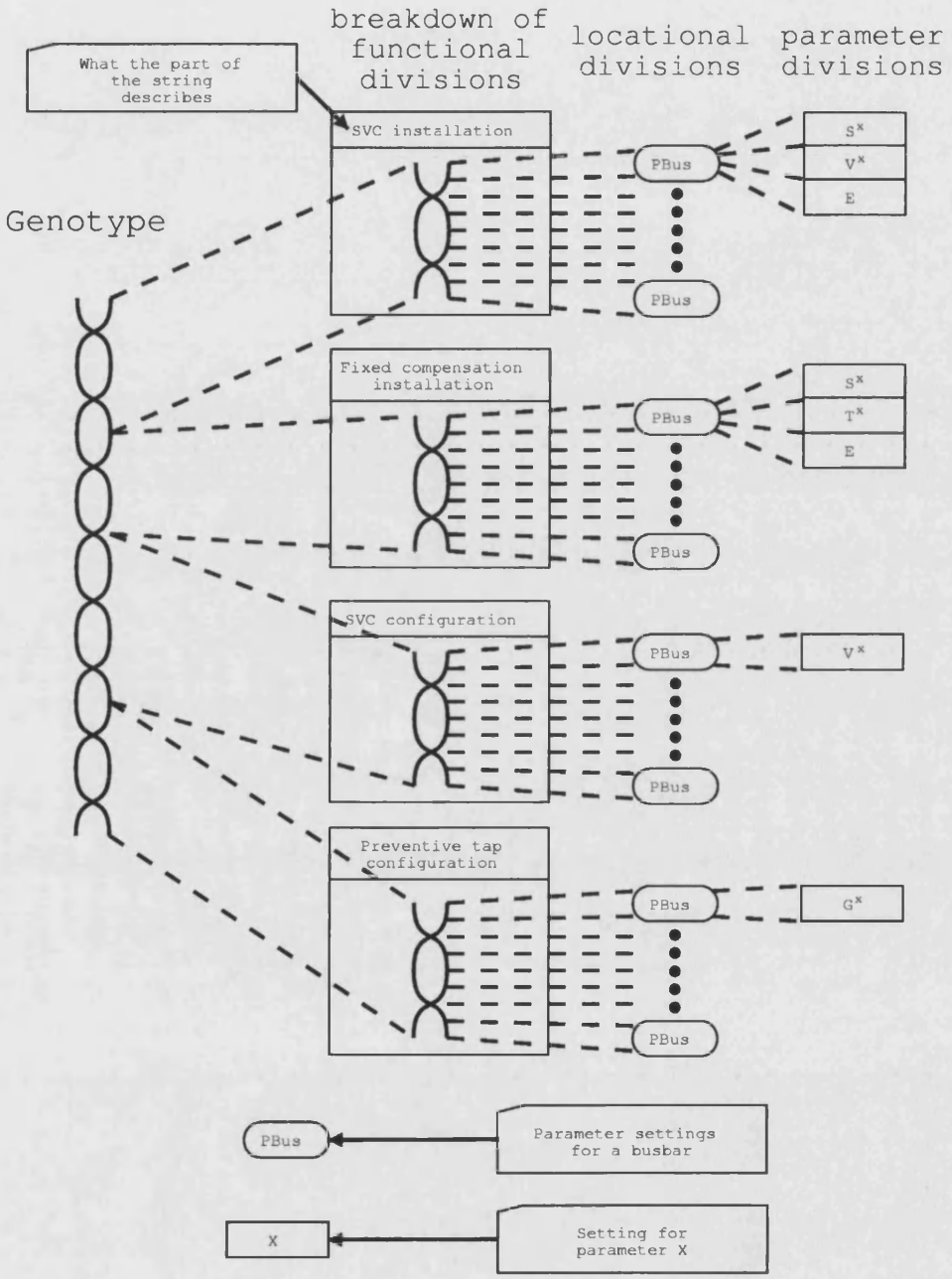

Figure 6.1: The complete string coding

Figure 6.1 shows how the string can be broken down in the functional blocks, and how these are then broken down into the parameter groups which apply to a particular busbar.

6.3 The Genetic Compensation Placement package

The following figures (6.3, 6.4 and 6.5) illustrate the design of the software in UML, an industry-standard language for specifying, visualising, constructing, and documenting the artifacts of software systems [Object Management Group, 2001], as introduced in section 6.1.1. The subset of UML required to interpret this figure is given in figure 6.2.

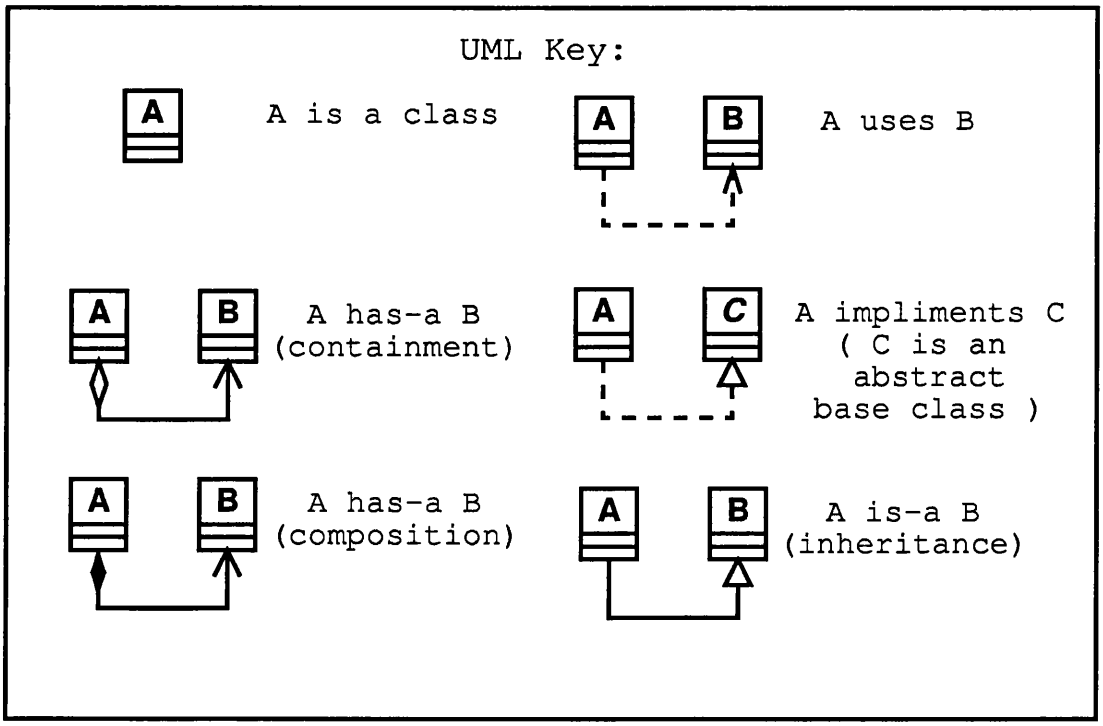

Figure 6.2: Guide to relevant Unified Modelling Language notation

From a top-level perspective the problem at hand requires four classes: a class to analyse and evaluate the performance of a candidate solution, a

genetic algorithm class to produce and optimise candidate solutions, a class to perform interfacing between the first two classes and evaluate the objective functions (based on the performance measures) and one to interface between the software and the user. Figure 6.3 shows the highest level structure of GCP. The figure should be interpreted as follows:

The user of GCP, represented by the stick figure at the bottom of figure 6.3, interacts with an object of the class `GeneticCompensationPlacement`; more concisely, they use an *instance* of `GeneticCompensationPlacement`. `GeneticCompensationPlacement` has instances of `GcpObjectiveFunction` and `GeneticAlgorithmEngine`. `GeneticAlgorithmEngine` uses an instance of `GcpObjectiveFunction`. `GcpObjectiveFunction` has an instance of `LoadFlowInterface`. What each class does will now be specified.

- `GeneticCompensationPlacement` uses a genetic algorithm to find optimal reactive power compensation siting schemes.
- `GeneticAlgorithmEngine` performs the mechanics of genetic algorithm optimisation using an external objective function to evaluate candidate solutions.
- `GcpObjectiveFunction` receives candidate solution strings (in this case, from `GeneticAlgorithmEngine`) and converts them into parameter lists of the format required by `LoadFlowInterface`. It then passes these solutions to `LoadFlowInterface` for analysis. It evaluates the results of the analysis performed by `LoadFlowInterface` and converts them into a set of non-correlating objective costs in the format required by `GeneticAlgorithmEngine`.
- `LoadFlowInterface` receives parameter lists which specify the parameters for the installation of new reactive sources and configuration of system plant. It implements these changes by modifying the power system study it has in memory and then performs load flow calculations on the

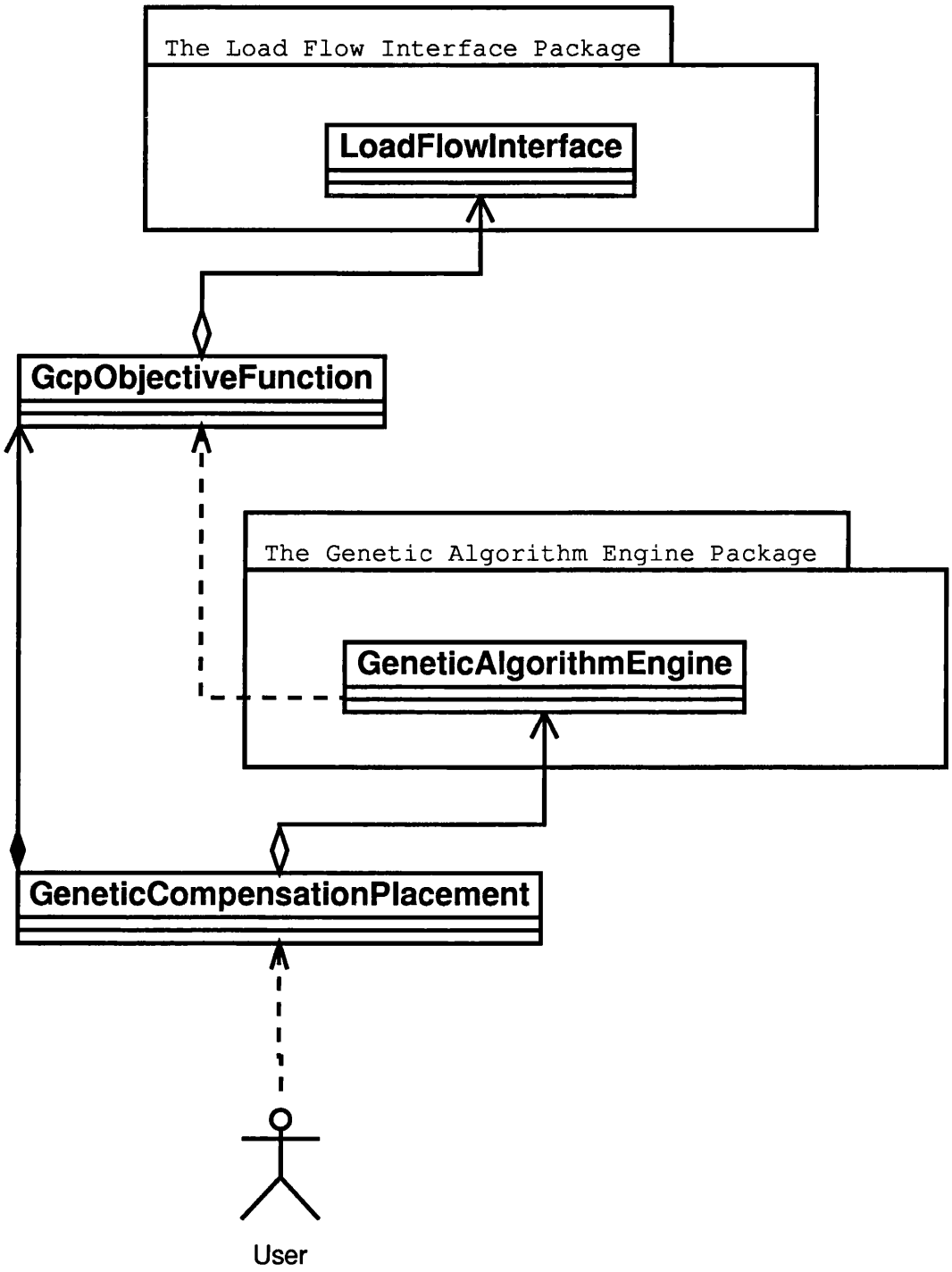

Figure 6.3: The hierarchy of GCP

study – this includes the base case and any contingencies. It analyses the results and returns a summary (in this case, to `GcpObjectiveFunction`).

Here one can begin to see the power of OOP. The `LoadFlowInterface` does all the work relevant to performing load flow, but the mechanics of how it does this is irrelevant outside of the class itself. Being a well designed class, all that `LoadFlowInterface` will present to the outside world is a small number of functions, which are all that is required to use the class. These functions are called its *interface*. Due to the fact that everything else is private, *encapsulation* is achieved: no external class will be able to access the internal workings and, more importantly, the internal data of `LoadFlowInterface`. This means that as long as the interface and functionality are maintained, any changes made within the class are guaranteed not to cause problems for any other class trying to use it. For example, because the internal representation of the data used by the class is hidden, one can be sure that no other class is relying on the data being contained in any particular structure (array or linked list, for example) or format (little endian or big endian, for example). Another benefit is that, because the private data cannot be accessed by any external class, these classes cannot accidentally damage the data managed by `LoadFlowInterface`. Encapsulation therefore maintains *data integrity*.

6.3.1 The objective-function object

As shown in figure 6.3, the `GcpObjectiveFunction` is the interface between the `GeneticAlgorithmEngine` and `LoadFlowInterface` objects. In a way, this is the object that implements the heart of the problem itself. It does this by offering two important functions: it takes candidate solutions, as are provided by the GA, and converts them into the parameter sets expected by the `LoadFlowInterface`, and it takes the performance data provided by `LoadFlowInterface` which it converts into a set of objective-function values, as required by `Gen-`

`eticAlgorithmEngine`. In effect, it is the only object in the software that knows that the problem being solved is the optimisation of a power system. This is a very important design consideration, as by separating the GA and power system sections of the program, their dependence on each other have also been separated: it would be impossible to modify one section and interfere with the operation of the other, except through the crudest means.

Note that, in chapter 8, which presents the results of the experiments carried out, results are presented for two different sets of objective functions – one set, for example, expresses the objectives used by National Grid. Each of these sets of objective functions required a different implementation of `GcpObjectiveFunction`. As each implementation provided the same interface and functionality they could easily be exchanged, thus modifying the specifics of the problem being solved, without the need for changes to be made elsewhere in the software. The implementations of `GcpObjectiveFunction` used are described in section 6.4.

6.3.2 The Genetic Algorithm Engine package

Initially the GA implemented by the *Genetic Algorithm Engine* (GAE) package was a Simple Genetic Algorithm of the type discussed in section 4.1. As will be discussed in section 6.4, this soon required extension so that it could optimise multi-objective problems. This was done through the use of Pareto optimality (section 4.4.3) and tournament selection (section 4.4.2). Further modifications were also made to the GA, as will be discussed in section 7.

The class hierarchy for the GAE package is given in figure 6.4. Again, the figure expresses the design in UML– see section 6.3 and figure 6.2.

The `GeneticAlgorithmEngine` class can again be seen and is the public interface to the package; it does little more than cause the genetic algorithm to start and stop optimising, and then present the optimal solutions found, as

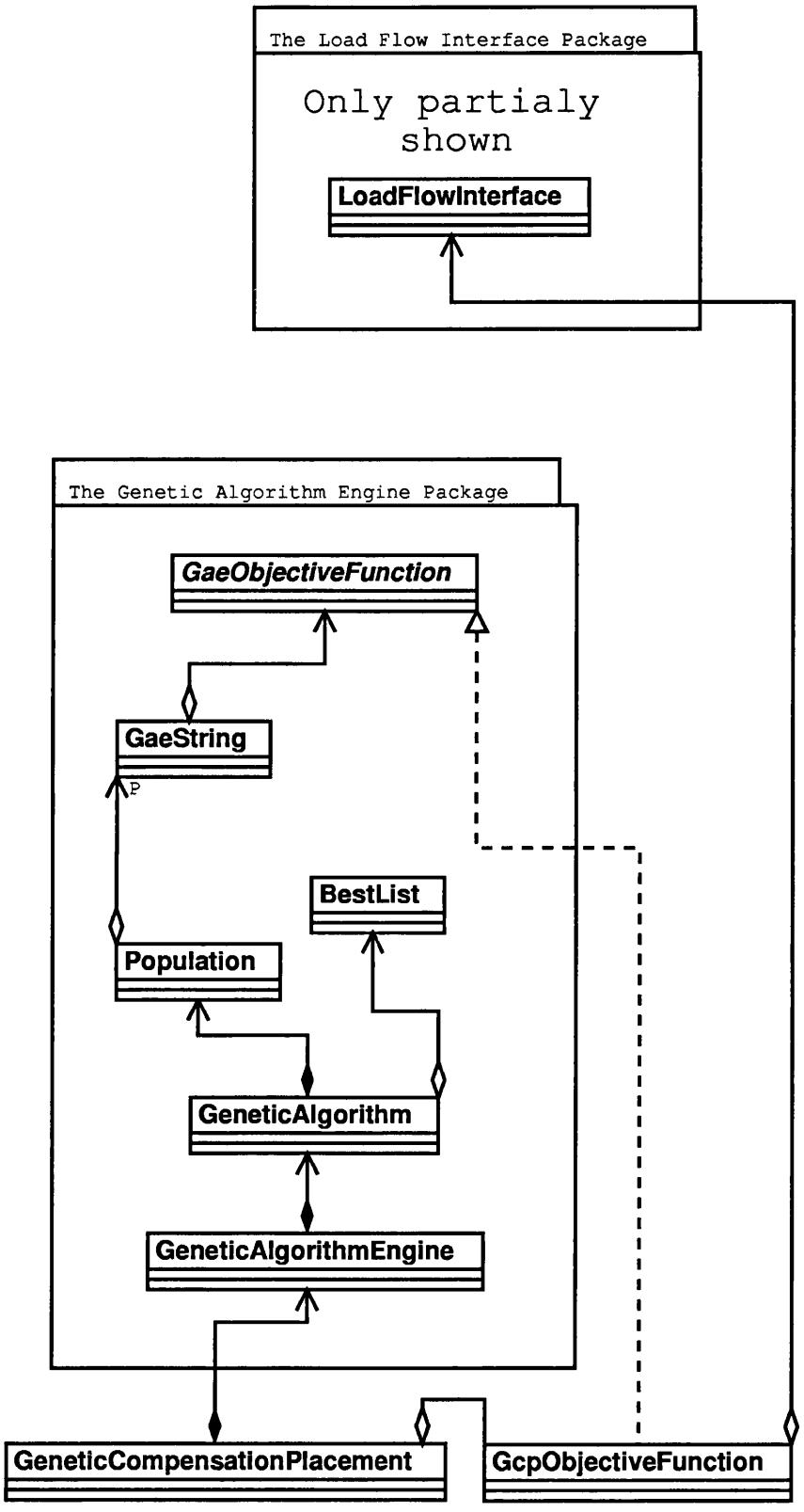

Figure 6.4: Class hierarchy of the Genetic Algorithm Engine package

requested. At this lower level of design the rest of the classes required can be seen. The design is fairly straight forward: the package interface, `GeneticAlgorithmEngine`, has an instance of `GeneticAlgorithm`; `GeneticAlgorithm` has an instance of `Population` and `BestList`; `Population` has instances of `GaeString`; and each `GaeString` has an instance of `GaeObjectiveFunction`.

The following list, discusses the functions of each class in detail.

- `GeneticAlgorithm`: called iteratively, it generates the next-generation population using genetic algorithm operators. Each time it is called it performs reproduction, crossover and mutation on the population – stored in an instance of `Population`. Each member of the population is then passed to an instance of `BestList` which maintains a list of the best solutions found.

- `Population`: a simple container class that contains instance of `GaeString`.

- `BestList`: a container that maintains a list of the best strings that are passed to it, using a Pareto comparison technique (see section 4.4.3). Note that for a single objective optimisation, only one solution can be the best – hence only one solution would ever be held by `BestList`– but for multi-objective optimisation, many solutions may be optimal.

- `GaeString`: stores the actual string value or *genotype* of a candidate solution. When requested – that is, when another class calls the relevant interface function – it returns the fitness of the candidate solution it contains. This fitness is enumerated by using an instance of `GaeObjectiveFunction`. For multi-objective optimisation, each string has the ability to indicate whether it is Pareto-dominates another string, that is, it can decide if it is "better than" another string.

- `GaeObjectiveFunction`: takes candidate solutions and evaluates their performance. It uses this performance data to enumerate a set of

objective-function values which it passes back to the calling object. This class is actually an *Abstract Base Class* (ABC): see the following discussion.

The existence of `GaeObjectiveFunction` seems strange: if `GcpObjective-Function` is the only class to implement the reactive power compensation objective functions but the GAE package uses `GaeObjectiveFunction` to evaluate the strings and GAE knows nothing of `LoadFlowInterface`, how does the correct evaluation of strings take place? The answer introduces another vital and powerful aspect of OOP: *inheritance*. If a class, `A`, inherits from another class, `B`, then `A` inherits all the functionality of `B` and can also extend it as necessary. This relationship is expressed using an *is-a* type relationship; `A` inheriting from `B` means `A` is a `B`. For example, there exist things called optimisation algorithms that find optimal solutions to given problems. A genetic algorithm does this, as does linear programming: a genetic algorithm *is an* optimisation algorithm and linear programming *is an* optimisation algorithm. Often, in the real world, it would only be important to know that an optimisation algorithm was being used, not which particular type. Within the world of OOP, a class could be made called `OptimsationAlgorithm` which would be an Abstract Base Type type from which `GeneticAlgorithm` would inherit. Thus, any class that knows how to use an `OptimsationAlgorithm`, would also be able to use a `GeneticAlgorithm`.

Back to the problem at hand, `GcpObjectiveFunction` inherits from `Gae-ObjectiveFunction`. When an instance of `GaeString` passes a candidate solution to `GaeObjectiveFunction` it is actually passing it to an instance of `GcpObjectiveFunction`, which to `GaeString` looks and behaves like `Gae-ObjectiveFunction`. As mentioned, `GaeObjectiveFunction` is an *Abstract Base Class* (ABC). It is abstract in the sense that it is more like the implementation of an abstract idea, rather than a usable class. Using the `OptimisationAlgorithm` class example, `OptimisationAlgorithm` would also be an ABC: you cannot

actually have an "optimisation algorithm" but you can have something that *is an* "optimisation algorithm".

Population stagnation time

As described, the `BestList` class maintains a list of the current optimal solutions found (or solutions, in the case of a multi-objective optimisation). Although it is indeed wise to ensure that these solutions are still represented within the population, there can be a great number of these optimal solutions, especially in the early stages of a multi-objective optimisation. Reinserting these solutions into the current population may cause an excessive loss of diversity: forcing the population to stay in the vicinity of previously optimal solutions may prevent new areas of investigation being found. To address this, GAE uses a measure called the *population stagnation time*, defined as the number of generations since a new optimal value was found. The current set of optimal solutions are not reinserted into the population until some predefined stagnation time can been exceeded (ten generations has been found to be fruitful). This gives the GA the opportunity for periods of non-profitable exploration without undue perturbation. If the population is found to have lost representation of the previously productive regions of the search space, the contents of `BestList` are substituted for the weakest members of the population, which will hopefully rejuvenate discovery of new optimal solutions..

6.3.3 The Load Flow Interface Package

`LoadFlowInterface` contains the public interface to the *Load Flow Interface* (LFI) package. It has four objects that it uses: `PowerSystemData`, `PlantModifierVector`, `ContingencySet` and `LoadFlowInterface`. The class hierarchy for the LFI package is given in figure 6.5.

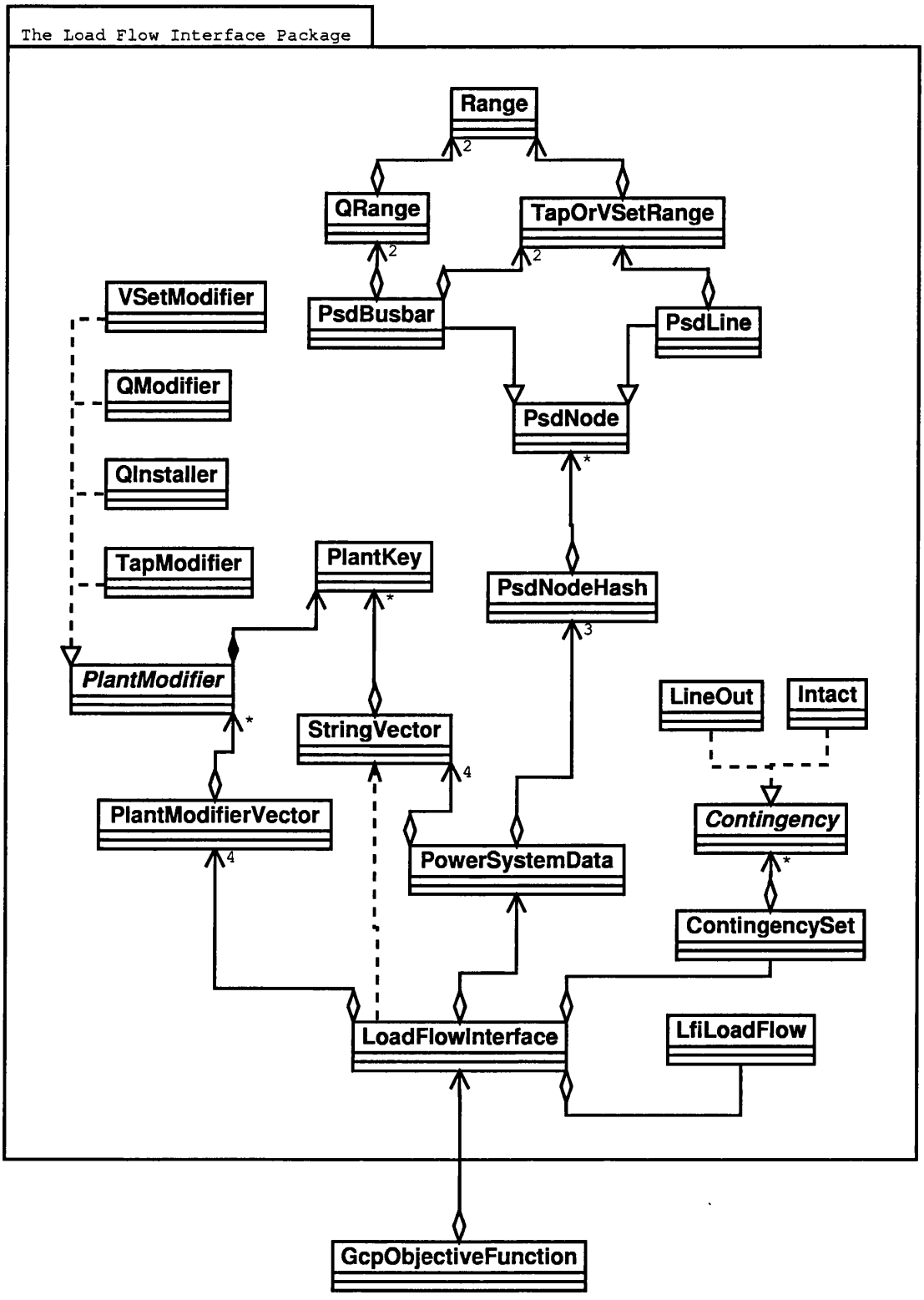

Figure 6.5: Class hierarchy of Load Flow Interface package

These classes are described in the following list.

- `PowerSystemData` stores data about a power system. Its public interface also facilitates a number of modifications to be done to the power system: these include setting transformer taps or transformer tap voltage targets, installing and configuring reactive power compensation and removing lines (simulating a line-out contingency). `PowerSystemData` can also read and write the details of a power system to a file.

- `ContingencySet` stores data about a contingency and can be used to apply these changes to an instance of `PowerSystemData`.

- `PlantModifierVector` stores data about how plant on a power system can be installed and modified. It takes sets of parameters and interprets them as details of specific changes to make. It then implements these changes by calling the required interface functions of a `PowerSystemData` object.

- `LfiLoadFlow` performs load flow calculations for a power system stored in a `PowerSystemData` object. It then analysis the results for each contingency – calculating the voltage deviation, power loss, real and reactive power usage, and VAr installation – and returns the resulting data to the caller.

The operation is best explained by an example:

After initialisation `LoadFlowInterface` will have the following objects: a `ContingencySet` for each contingency, a `PowerSystemData` object which holds details of the power system, and a `PlantModifierVector` which holds details for all the plant which may be affected by the class user. In this case, that user is actually an instance of `GcpObjectiveFunction`.

When `LoadFlowInterface` is called with a new parameter set it creates a copy of its `PowerSystemData` object. This new copy of the power system data is then passed to the `PlantModifierVector` along with the set of parameters. The `PlantModifierVector` interprets the parameter set as a set of changes to make to the power system data: tap settings, voltage set points and new reactive power compensation to install. On return, `LoadFlowInterface` receives the data representing the base case with the requested changes made.

The contingent cases then need to be made. This is done by making copies of the modified `PowerSystemData`: one for each contingency. Each of these copies, in turn, are processed by an instance of `ContingencySet` which calls the required functions of `PowerSystemData` to cause the changes required to implement the contingency to be made.

`LoadFlowInterface` now has the compete set of modified cases that make up the study with the requested modifications: tap settings, installation, and so on. Theses cases can then be, in turn, passed to the instance of `LfiLoadFlow` which performs the load flow calculations and returns a summary of the results. After each case has been processed, `LoadFlowInterface` has a set of load flow summaries which can be passed back to the caller – `GcpObjectiveFunction` in this case – thus completing its task.

6.3.4 Third party load flow calculation packages

Section 3.3 discussed how Bridenbaugh *et al* [1992] introduced the use of Optimal Power Flow (OPF) software so that control of auxiliary system plant – corrective taps, and so on – could be moved into a sub-problem. There are many OPF implementations available, so rewriting a load flow package was an unnecessary expense. GCP uses a third party load flow package, accessed though the `LfiLoadFlow` class.

A number of implementations of OPF were tried. Initially, the package

called *Complexe Power Flow* (CPF) was used. CPF was developed by the PSMAC group at the University of Bath [Dunn, 2001]. As this was an "in house" package the source code was available. This meant that CPF could be included in GCP at compile time, which provides the fastest interface for communication. Unfortunately, this implementation caused significant problems. CPF is written in the C programming language, which is not object oriented, and was not ever intended to perform load flow calculations for more than one case: CPF was intended to be run once and finish. It therefore did not free any memory it used. If it had been written in an object oriented language, after each set of load flow calculations the *instance* of CPF's calculation core could have been deleted, freeing the memory. This was not the case and significant problems were found in trying to manage the memory used by CPF.

Eventually this approach was abandoned and CPF was used as an external stand alone program that GCP called each time it required load flow. As discussed in section 6.1.2, all files handled by GCP were in XML. A program was therefore required and written to convert between XML and CPF's native file format. It is hoped that in the future other developers will use this XML interface to CPF. When, during testing, real world problems where considered it was found that CPF did not adequately handle corrective taps. As discussed, corrective taps attempt to maintain the busbar voltage by automatically adjusting their tap setting and CPF did not support this. In the end a package called OPFL02 was used. This is the load flow software used by National Grid, and so it has the added advantage that conformance with National Grid study systems is guaranteed. It was only available as a binary executable, but as GCP already supported external load flow packages at this point, all that was required was a modified version of the XML-CPF conversion interface that supported the OPFL02 file format.

6.4 Objective functions

Two objective function sets were used by GCP, and results for each can be found in chapter 8. The first (section 6.4.1) was for comparison with SCORPION, and the second was an improved formulation, which attempted to find solutions that reduced the stress on the system states by using the added objective of voltage profile optimisation.

6.4.1 The reactive power cost minimisation objective set for SCORPION comparison

As discussed earlier, to rigorously compare the performance of GCP to that of SCORPION, National Grid's RCP software that was introduced in section 3.5, both packages must be trying to solve the same problem. SCORPION attempts to minimise the reactive power cost, as discussed in section 5.6.2. Therefore, for SCORPION comparison the objective is, primarily:

$$\max_{\varsigma} \quad \theta_{RPC}(\varsigma, \sigma)$$

This objective was found to perform properly when improving the performance of convergent systems, but a weakness was noted. When the study systems – the base case and contingent systems – were initially non-convergent, the likelihood of the GA finding convergent solutions was unacceptable low. This was because the objective function provided no measure of failure. This situation was improved by additionally using the busbar violation measure discussed in section 5.6.3:

$$\max_{\varsigma} \quad \theta_{\Upsilon}(\varsigma, \sigma)$$

These two objective functions were combined using, firstly, a weighted sum approach and, secondly, by implementing a multi-objective approach. The multi-objective approach was found to perform significantly better in that it found convergent solutions for all studies considered, including those with highly stressed initial system states, and, therefore, the weighted sum approach was discontinued.

To further encourage convergent solutions, the voltage deviation objective, as discussed in section 5.6.1, was also incorporated:

$$\max_{\varsigma} \quad \theta_{Vdev}(\varsigma, \sigma)$$

Note that δv_b, the deviation tolerance, was set to zero. Inclusion of this measure produced an interesting effect: although convergent system were more likely to be found, the best solutions found in the search were of a higher reactive power cost than those found without the inclusion of the voltage deviation objective. This is because once convergent solutions have been found the search effort becomes divided between minimising reactive power cost and voltage deviation. This was overcome by using *objective switching*. As the number-of-limit-violating-busbars measure dictates a convergent solution, it can be used to control whether or not voltage deviation is considered. When the number of violating busbars is zero the voltage deviation is also set to zero. Voltage deviation is, therefore, used only as a measure of failure and not as an objective to be optimised. Formally, this objective is expressed as follows:

$$\max_{\varsigma} \theta_{Vdev}(\varsigma) = \begin{cases} 0 & \text{if} \quad \theta_{\Upsilon} = 0 \\ -1 \times \rho_{Vdev} & \text{if} \quad \theta_{\Upsilon} \neq 0 \end{cases} \tag{6.1}$$

Using some basic heuristics, another approach to reducing the reactive power cost was also tried. The salient idea is a follows: if there are less MVAr sources

installed, the reactive power cost will not only be significantly reduced, but there will also be less VAr sources on the system to provide chargeable VArs. To specifically encourage less VAr installation, in addition to already being included in the reactive power cost, another objective was added that just expressed the MVAr installation:

$$\max_{\varsigma} \quad \theta_{Inst}(\varsigma, \sigma)$$

This equation was discussed in section 5.6.2.

6.4.2 Reactive power cost minimisation and optimal voltage profile objective set

The second objective set to be considered was basically the same as the SCORPION objective set but the voltage deviation was kept as a permanent objective of the optimisation, and not dependant on θ_{Υ}. This was formulated after discussions with National Grid [Aldridge *et al*, 2001] who suggested that, although not currently done by SCORPION, a planning tool that found solutions to the reactive power planning problem that were optimal in terms of both reactive power cost and voltage deviation would be preferential.

Chapter Seven

Improving the performance of GCP

This chapter introduces a number of techniques that were used to improve the performance of the GA used by GCP. After these techniques have been presented, two GA based algorithms are specified which utilise these techniques, each at an increasing level of sophistication. Both of these algorithms, in addition to an SGA, have been implemented in the course of this work.

7.1 Integer coding

An area that can be explored in the search for improving the performance of the GA is the string representation. Previously a binary coding method has been employed, but an integer coding scheme is also popular [Mantovani *et al*, 2001; Deb and Beyer, 2001]. An integer coded GA uses a string of integers to represent the genotype; each integer specifies the value of one parameter value, that is, one identifiable unit of the solution. An example of such a parameter would be the size of a device or a particular tap setting.

The strings are subjected to the same operations as they are in SGA optimisation, but due to the fact that they are now operating on integers means that they must be slightly modified. Instead of mutation switching bits between one and zero, an integer parameter value that has been selected for mutation is changed to a randomly chosen replacement value. As before, crossover cuts strings at some randomly selected point, between two units of the string, and swaps the "tails" of the strings, but now the strings can only be cut between consecutive integers, as opposed to being cut between consecutive bits.

7.2 Parameter templates

A genetic algorithm's convergence can be affected by the solution encoding, which can cause problem space distortion. This was pointed out in section 4.2 and, basically, revolves around the fact that when a parameter is encoded within a binary number the number of different values that the binary number can take is rarely equal to the number of values the parameter can take. This means that the genotype can specify a value for a parameter that is out of the legal range. This value must then be, for example, rounded down to an acceptable value. This would cause a distortion in the problem space. A more detailed analysis of this effect, as well as other correction schemes were described in section 4.2. Unfortunately, all schemes result in undesirable effects. This section describes a technique which aims to alleviate this effect.

There are three causes of new information within the strings of the GA: initialisation, crossover, and mutation. Within the first generations, a major cause of illegal parameter values is initialisation – the act of generating the first population by using a random number generator. The problem at this stage is that the GA has no information about what values the genotype may take, and so the random numbers are constrained only by the capacity of the sub-strings themselves. This can be overcome by using a parameter template set.

As already discussed, there exists a path through the hierarchy of GCP for the purposes of evaluating candidate strings. Candidate strings are passed from the GA up through the hierarchy, via the objective function, to the load flow interface where they are implemented and evaluated. Performance data then flows back down through the hierarchy to the GA where it is used by the genetic operators.

Another path is added to the hierarchy which shadows this "string evaluation" route, but this new path is used before optimisation is started, for the

purpose of gathering information about the problem being solved. Before the GA creates the initial population it sends out a request, through the hierarchy, for a *parameter template*. This template is identical to the integer representation of a candidate solution, but each "parameter value" is actually the maximum value that that parameter can take. The GA then uses this data during initialisation and optimisation to prevent illegal values occuring in candidate solutions.

7.3 Simulated length for mutation-point and crossover-point selection

Integer coding represents each parameter with a sub-string of length one. This looses an aspect of a binary representation in which the sub-string length is a function of the number of possibilities that the parameter encodes. This means that any GA operator that affects the string at a particular position – and this position is chosen as a function of the string length – inherently uses this information about the number of possiblities each parameter may take. Once the maximum legal value for a particular parameter is available, a technique can be implemented for an integer representation by considering the *information content* of each parameter. This section looks at this effect and the proposed algorithm which has been developed in more detail.

Through parameter templates, the information about the maximum value for each integer parameter is available, and it can be put to use in the other genetic operators. One aspect of a binary coded scheme is that the number of discrete values that a sub-string encodes is proportional to its length. This is a factor in any operator that operates at a particular point along the string. Mutation is a good example of this. The position along the string that mutation affects is chosen by evaluating a random number between one and the number of bits in the string: the string length.

Consider a string that encodes two independent parameters. The first, parameter A, has only two possible values and the other, parameter B, has four possibilities, for example. Using a binary encoding scheme, parameter A would be represented by just one bit and therefore a sub-string length of one, while parameter B requires two bits and hence has a sub-string length of two. With this representation, B is twice as likely to be mutated in some way than A. This effect is lost by using an integer representation: both sub-strings would be of length one. The same effect also comes into play with crossover when choosing the crossover point along a string.

This concept of sub-string length may well be important to the search ability of the GA, as the search effort is distributed according to the number of possibilities encoded within a sub-string. Although at this generalised level one cannot say that one sub-string or parameter is more important to the optimisation process than another, a higher number of encoded possibilities would suggest that more effort is required to find the best value for that parameter.

With the information now available from the parameter templates, this effect can not only be retained, but improved upon. Following on from the previous example, if parameter A had, instead, covered sixteen possible values and B covered seventeen possible values then the smallest possible binary encoded sub-string lengths would be four and five, respectively. This information could easily be calculated using the template data and used during length based operations, but as can be seen, a significantly longer string length is used for B than for A compared to the relatively small difference in the number of possibilities they encode. This is due to the drawback that binary sub-strings have to be an *integer* number of bits long. When using integer coding, the bit-length of each sub-string is only an abstract concept; hence no such constraint is in effect.

The proposed algorithm is to formulate a meta-string that is structurally the

same as the parameter templates, but stores the maximum *entropy* of each parameter. In terms of abstract string length – which is*not* constrained to be an integer value – each parameter's length is equal to its information content. In the previous example, parameter A would have an effective length of $\log_2 16 = 4$ bits and parameter B would have an effective length of $\log_2 17 \approx 4.09$ bits. The information content of the entire string is then used as the effective length of a string. Then, instead of choosing a point along the string using a random number between one and the string length, a floating-point random number is chosen between one and the effective length of the meta string. The actual position is chosen by finding the required position on the meta string and calculating the associated parameter on the actual string. Theses concepts of string length are further illustrated in figures 7.1 to 7.3.

Figure 7.1 shows a representation of an example integer coded string. In the example, there are seven parameters encoded by the string: these are labelled P1 thorough to P7. As can be seen, the string is seven integers long. Below the string is the associated parameter template. From this it can be seen that P1 can be one of two different values – for example, zero or one – P2 can be one of five different values, P3, three and so on. Figure 7.2 shows how this would be structured if a binary representation was used. The string is now a string of sixteen bits. The length of each sub-string is given in table 7.1 in the column marked "Bits required".

This table also shows, in the column marked "Information", the actual entropy of each parameter. This is the data used to formulate the meta string shown in figure 7.3. The proposed algorithm would use this meta-string to convert between a virtual length measurement and a particular position along an actual string. For example, a position of nine and a half bits along the string corresponds to parameter P6. As can be seen, the meta-string is shorter than the binary encoded string – 13.399 units compared to sixteen – because no redundancy exists in the meta-string representation.

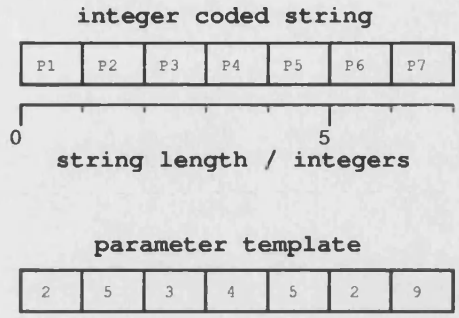

Figure 7.1: Example of an integer representation of a string with associated parameter template

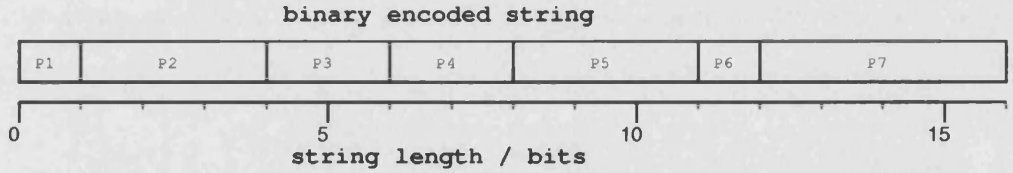

Figure 7.2: Structure of a binary representation of the example string

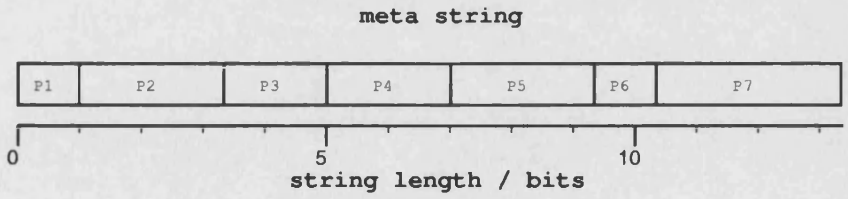

Figure 7.3: The structure of the meta-string

Table 7.1: Example of formulating a meta-string

Parameter	Number of possible values	Information	Bits required
P1	2	1	1
P2	5	2.321	3
P3	3	1.585	2
P4	4	2	2
P5	5	2.322	3
P6	2	1	1
P7	9	3.170	4
	Total	13.399	16

This functionality was added to GCP by implementing a new class, `CodingTransformer`, that is used by `GeneticAlgorithmEngine` to convert between simulated lengths and real lengths.

7.4 Simulated binary crossover

Crossover, the genetic operator, is essential to the GA search because it can combine favourable variations in a number of different strings into one hybrid string. When an integer coded string is used, however, it is found that the standard crossover algorithm does not create the same amount of diversity as it does with a binary representation, and is therefore less powerful. This is because crossover operates at points *between* the digits of the string; the two parents are then cut at this point and their tails swapped. With a binary representation the crossover point could be between any two consecutive bits, which means that the crossover point could be between two sub-strings or even at some point within a sub-string. This creates a greater diversity of

offspring than with an integer representation where each integer *is* a sub-string. Hence crossover can only act at a point between sub-strings.

What is needed is a technique to simulate crossover acting within a sub-string when that sub-string is an integer; as will be discussed, many such crossover techniques have been developed. These techniques allow the crossover point to act at a point within an integer, for some point i along the parent strings. For example, with two strings at time t, $X^{(1,t)}$ and $X^{(2,t)}$, composed of I sub-strings:

$$X^{(1,t)} = x_1^{(1,t)}, x_2^{(1,t)}, \ldots, \quad x_i^{(1,t)}, \quad \ldots, x_{I-1}^{(1,t)}, x_I^{(1,t)} \tag{7.1}$$

$$X^{(2,t)} = x_1^{(2,t)}, x_2^{(2,t)}, \ldots, \quad x_i^{(2,t)}, \quad \ldots, x_{I-1}^{(2,t)}, x_I^{(2,t)} \tag{7.2}$$

$$\uparrow$$

crossover point

To create the two children, at time $t+1$, first the tails of the strings (the sub-strings from $i+1$ to I) can easily be crossed over:

$$X^{(1,t+1)} = x_1^{(1,t)}, x_2^{(1,t)}, \ldots, \quad x_i^{(1,t+1)}, \quad \ldots, x_{I-1}^{(2,t)}, x_I^{(2,t)} \tag{7.3}$$

$$X^{(2,t+1)} = x_1^{(2,t)}, x_2^{(2,t)}, \ldots, \quad x_i^{(2,t+2)}, \quad \ldots, x_{I-1}^{(1,t)}, x_I^{(1,t)} \tag{7.4}$$

$$\uparrow$$

crossover point

The problem is what to do at the crossover point it self. That is, how should $x_i^{(1,t+1)}$, and $x_i^{(2,t+1)}$ be evaluated?

As a simple example, a scheme could be adopted in which, at the crossover

point in both strings, the average value at the crossover point is used:

$$X^{(1,t+1)} = x_1^{(1,t)},\ x_2^{(1,t)},\ \ldots,\ \frac{x_i^{(1,t)} + x_i^{(2,t)}}{2},\ \ldots,\ x_{I-1}^{(2,t)},\ x_I^{(2,t)} \tag{7.5}$$

$$X^{(2,t+1)} = x_1^{(2,t)},\ x_2^{(2,t)},\ \ldots,\ \frac{x_i^{(2,t)} + x_i^{(1,t)}}{2},\ \ldots,\ x_{I-1}^{(1,t)},\ x_I^{(1,t)} \tag{7.6}$$

↑

crossover point

Such a technique, however, is not commendable as its cumulative effect will be that all values will tend towards the middle of their range. Fortunately, more sophisticated techniques have been developed. Deb and Beyer [2001] presented a synopsis of the leading crossover algorithms for real-parameter coded GAs; a summary is presented in the following sections.

Linear operators

A linear operator can be used to create three solutions, $(x_i^{(1,t)} + x_i^{(2,t)})$, $(1.5x_i^{(1,t)} - 0.5x_i^{(2,t)})$, and $(-0.5x_i^{(1,t)} + 1.5x_i^{(2,t)})$ from the parent sub-strings $x_i^{(1,t)}$ and $x_i^{(2,t)}$. This then provides three possible children. The algorithm intimates that the best two children should be selected.

Blended operators

The *blend crossover operation* (BLX-α) takes the two parent sub-strings $x_i^{(1,t)}$ and $x_i^{(2,t)}$ (assuming $x_i^{(1,t)} < x_i^{(2,t)}$) and selects the two solutions from the range $[x_i^{(1,t)} - \alpha(x_i^{(2,t)} - x_i^{(1,t)}), x_i^{(2,t)} + \alpha(x_i^{(2,t)} - x_i^{(1,t)})]$. This is done by selecting a

random number, u_i, in the range zero to one and using

$$x_i^{(1,t+1)} = (1 - \gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)} \tag{7.7}$$

$$x_i^{(2,t+1)} = (1 - \gamma_i)x_i^{(2,t)} + \gamma_i x_i^{(1,t)} \tag{7.8}$$

where

$$\gamma_i = (1 + 2\alpha)u_i - \alpha \tag{7.9}$$

The value α is a constant that can be used to tune the effect of the BLX operator. A value of 0.6 (referred to as a BLX-0.6 operator) has been found to be suitable.

Unimodal normally distributed crossover operator

The unimodal normally distributed crossover operator uses three parents to produce two or more children. Children are selected from an ellipsoidal probability distribution with one axis formed along the line joining two of the three parent solutions, and the extent of the orthogonal direction is decided by the perpendicular distance of the third parent from the y axis.

Simulated binary crossover operator

Deb and Beyer [2001] conclude that the BLX operator has the greatest potential for learning as the location of child solutions depends on the difference in parent solutions. Other proposed techniques have either a uniform distribution (linear operators) or have a probability distribution that tends to place children in the centre of the parent solutions (unimodal normally distributed crossover). With the BLX operator, however, if the difference between the parent solutions is small, the difference between the child and parent solutions if also small. They state: "We believe this is an essential property for any search

algorithm to exhibit self adaptation. This is because the spread of the current population dictates the spread of solutions in the resultant population."

Deb and Beyer then proposed their own crossover algorithm that exhibits this characteristic: the *Simulated Binary Crossover* (SBX) operator. The SBX operator simulates the working principal of the single point crossover operator on binary strings. The operator respects the interval schemata processing in the sense that common interval schemata between parents are preserved in children.

Again, the notation that two child solutions $x_i^{(1,t+1)}$ and $x_i^{(2,t+1)}$ are being equated from two parent solutions $x_i^{(1,t)}$ and $x_i^{(2,t)}$ will be used. The algorithm, which uses η to tune the probability distribution, is as follows:

- **Step 1:** choose a random number u_i such that $0 \leq u_i \leq 1$
- **Step 2:** calculate

$$\beta_{qi} = \begin{cases} (2u_i)^{\frac{1}{\eta+1}} & : \text{ if } u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta+1}} & : \text{ otherwise} \end{cases} \tag{7.10}$$

- **Step 3:** calculate children using

$$x_i^{(1,t+1)} = 0.5[(1+\beta_{qi})x_i^{(1,t)} + (1-\beta_{qi})x_i^{(2,t)}] \tag{7.11}$$

$$x_i^{(2,t+1)} = 0.5[(1-\beta_{qi})x_i^{(1,t)} + (1+\beta_{qi})x_i^{(2,t)}] \tag{7.12}$$

Two aspects of this technique particularly deserve highlighting. Firstly, the children solutions are symmetric about the parents; there is no biasing towards any parent in a single crossover operation. Secondly, for a fixed η, the spread of children solutions is proportional to that of the parent solutions, with β_{qi} as the constant of proportionality.

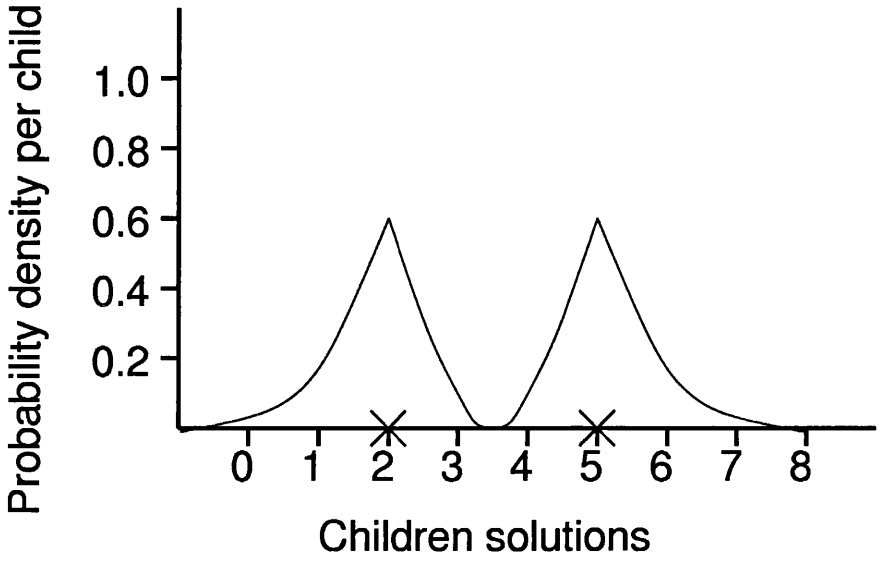

Figure 7.4: Probability distribution for creating solutions of continuous variables, from parents $x_i^{(1,t)} = 2$ and $x_i^{(2,t)} = 5$, using the SBX operator with $\eta = 2$

An example of a probability distribution is given in figure 7.4.

The distribution shown is that of the probability for the generation of children from the parents $x_i^{(1,t)} = 2$ and $x_i^{(2,t)} = 5$ with $\eta = 2$. The symmetry of the distribution can be seen. Depending of the value of u_i, the pairs of children could be, for example, 1 and 6 or 3 and 4.

Note that figure 7.4, illustrates the distribution with continuous variables. If continuous variables are not used the children will require rounding of some sort. Note that in addition, parameter templates – as discussed in section 7.2 – can also be used to limit the resultant values to the legal range.

7.5 The protein coding analogy

Even when concentrating on the details of something, it is always important to remember what is actually trying to be achieved. In this case it is an optimisation algorithm based on natural genetics and evolution. With

the previous modifications in mind, the question may be asked, "does the algorithm still fit within this model". Returning to genetic theory an aspect arises that has hitherto not been considered: proteins. The SGA models the lowest level of genetics – the genotype – with implemented solutions being the observed phenotype. Genes are perpetuated as sequences of nucleic acid, but functions by being expressed in the form of proteins [Lewin, 1997]. One can think of the relationship as being "DNA encodes the instructions for making, for example, a human being, but proteins are what actually go and make one". Proteins represent another level of abstraction, that lies between that of the genotype and phenotype. If the previous modifications are implemented, a string is taking on more of an analogy with a protein sequence than the conventional nucleotide sequence of the SGA.

Investigation into how DNA and protein relate reveals the following. DNA does nothing but act as a reference; *mRNA* represents the "working copy" of DNA, where the "m" stands for *messenger*. mRNA is actually a sequence of nucleotides, which are grouped into threes; a nucleotide triplet is a *codon*, of which there are 64. Of these 64, 61 of these codons encode for amino acids, and proteins are sequences of these amino acids. The remaining three codons are stop codons that terminate amino acid sequences. The particularly interesting thing in this process is how nucleotide sequences code for the amino acids. This mapping is known as *the genetic code* and reveals some interesting properties. The genetic code is unambiguous, degenerate and almost completely universal. Each codon corresponds to only one amino acid: unambiguous. There are 20 amino acids and most of the amino acids correspond to several codons: degenerate. Finally, codons with similar sequences specify chemically similar amino acids – hence structurally similar proteins [Rawn, 1989]. All these factors combined result in the following quality: many mutations have no effect; of the remainder, most will result in a protein with similar behaviour, with the minority producing quite different protiens.

7.5.1 Protein mutation

Qualitatively, these benefits can be implemented within an integer coded GA by using three flavours of mutation. The first of these has no effect so can be disregarded – and is mentioned only for completeness – the second two are slight and severe. As proteins are represented by integers, a slight mutation can be implemented as a small increase or decrease in the value. Severe mutations can be implemented by a large change or by simply replacing the parameter with a randomly chosen replacement. Once a location for mutation has been chosen, the type of mutation – that is, slight or severe – can be randomly decided and carried out.

7.6 Protein coded GA

An implementation of a protein coded GA was tried. The GA uses integer coding (section 7.1), parameter templates (section 7.2), simulated length for mutation-point and crossover-point selection (section 7.3), protein mutation (section 7.5.1) and simulated binary crossover (section 7.4).

7.7 Variable location siting

It must be pointed out that there is a completely different approach to one aspect of the formulation of solution strings: how locations of devices to be installed are represented within a solution string.

Up to this point, the location of each reactive power compensation device was decided by the index of the sub-strings that described it. Remember that, in the case of an SVC, a device is described by sets of sub-strings: the size, the voltage set-point and an install/don't install flag. In the previous coding scheme, the first set of SVC sub-strings described the SVC to be installed at

the first candidate busbar, the second set of sub-strings, the second candidate busbar and so on. An alternative representation is one that also encodes the location at which the devices should be sited *within* the string. For each SVC there would therefore be an extra sub-string. The first three would be the size, voltage set-point and install flag as before, but with an extra sub-string that encoded the location. Now, when the string is decoded, the first set of SVC sub-strings would describe an SVC to be installed and at which location to install it – for example, the first set of sub-strings could describe an SVC to install at the fifth candidate busbar.

Such a scheme was implemented by the author of this work [Pilgrim and Li, 2000b,a] and found to be adequate for small test problems, but was inadequate for larger systems. There are two reasons why such a scheme does not perform well. Firstly, the technique allows more than one SVC to be sited at the same node. Although this is not necessarily bad, it makes it extremely difficult to implement when there are expansion constraints: constraints on the amount of compensation that may be sited at any particular candidate busbar. Another reason becomes apparent when a situation is considered in which the system requires only one device to be sited at a certain busbar: the GA has to try to find the correct busbar-number at which to site the device. Because the actual busbar numbers are not meaningful to the problem – for example, candidate busbar one may not be anywhere near candidate busbar two - the problem is now becoming a *needle in a haystack* problem which is classically difficult for a GA to solve (see section 4.5 for more discussion on this topic.)

7.8 Genetic Algorithms used

Three GAs have been implemented in this work. These are as follows:

- **Simple Genetic Algorithm: SGA.** This is the SGA as described in section 4.1

- **Integer coded Genetic Algorithm: IGA.** This is a GA that uses a single integer value to encode each sub-string, as opposed to the binary bit string used to encode each sub-string of the SGA. The algorithm also uses parameter templates for initialisation and mutation (section 7.2).
- **Protein coded Genetic Algorithm: PGA.** This is the IGA with all of the modifications described in section 7.6.

The specific differences are summarised in the following table.

Table 7.2: The differences between the GA implementations used in this work

	SGA	IGA	PGA
Binary coded sub-strings	⋆		
Integer coded sub-strings		⋆	⋆
Parameter templates		⋆	⋆
Entropy length for crossover-point and mutation-point selection			⋆
Protein mutation			⋆
Simulated Binary Crossover (SBX)			⋆

Chapter 8 presents results for each of these three algorithms, as named in table 7.2.

7.9 GA variables

Finally, another technique for improving the performance of GAs is via optimisation of the GA parameters. The GA parameters that are considered in this work are summarised as follows:

- Mutation rate.

  The probability of mutation occurring.

- Crossover probability.

  The probability of crossover occurring between two parents.

- Reproduction rate.

  The fraction of the population which participates in the reproduction process.

- Population size.

  The number of candidates solutions contained within a population.

- Number of generations.

  The maximum number of generations that the GA is permitted to run for. In the general field of optimisation, this is the same as the number of iterations for which the algorithm runs for.

Chapter 8 investigates the effects of varying these parameters of the GAs used.

Chapter Eight

Results

The chapter presents analyses of the performance of GCP and a comparison with a linear programming technique.

In section 8.1, the test problem used is discussed. The problem was developed, with the cooperation of National Grid, to represent a practical sized problem. The study considered is named the Beta study. The Beta study is based on a practical-size study call the Alpha study, but with the exception that three of the generators have been disabled and there five cases, four of which represent line-out contingencies. This has been done to increase the requirement for reactive power compensation and, thus, to provided more revealing results.

Section 8.2 concentrates on improving the performance of GCP by optimising the search algorithm employed. The Beta study is used in all experiments. Results are presented for the three GAs, defined in section 7.8, using a number of different crossover rates, reproduction rates, mutation rates, population sizes, and trial lengths. The section concludes with the specification of the two GAs that were found to produce optimal results. The first of which is chosen to be the GA used by GCP in further analyses.

Section 8.3 presents a comparison of a solution found by GCP with the solution found by National Grid's SCORPION package, which uses a linear programming based technique. Both techniques are tested on the Beta problem, and are attempting to minimise the reactive power cost without violating any operational or system constraints, as defined in section 6.4.1. GCP is shown to outperform SCORPION by finding cheaper solutions to the problem.

Solutions generated by GCP using the cost reduction objective are found to posses an undesirable characteristic. Although generated solutions maintain all busbars voltages within the system constraints, in the search for minimum-cost solutions, the system states can become highly stressed, with many busbar voltages approaching their limits. Although these are valid solutions within the context of the problem, they raise concerns about the security of the power system under unforeseen contingencies. To address this, GCP is extended to optimise a fully multi-objective problem: minimise reactive power cost and minimise voltage deviation; previously, voltage deviation merely needed to be within the associated limit.

Section 8.4 presents an analysis of the solutions found by the multi-objective optimisation. The set of Pareto-dominant solutions found are shown to significantly improve the voltage profile at an increased cost. These solutions are again compared to SCORPION's solution, and it is found that GCP can find cheaper solutions than SCORPION, which significantly improve system performance.

8.1 Practical test system: the Beta study

An important aspect of this work has been to develop a solution to the reactive power compensation planning problem that can successfully optimise real world problem. For the purposes of validation, a practical test system was developed with the cooperation of National Grid. A brief summary of the system is presented in table 8.1.

The seven generators on LV (low voltage) circuits are connected to the Supergrid via the five preventive transformer taps. These taps must be optimised, according to section 5.3. For the actual installation of compensation there are fifty five busbars that are candidates for the installation of fixed capacitors (FCs) and five busbars that are candidates for SVC expansion. FC

Table 8.1: A summary of the test system

Busbars	68
Lines	141
Generators (LV)	9 (7)
Preventive transformer taps	5
FC candidate busbars	55
expandable SVCs	5

candidate busbars may have one or more sixty MVAr capacitors installed. The five locations that are candidates for additional SVCs may have units installed in the sizes specified in table 5.1. Even if no SVC is installed at one of these sites, the voltage set point of the existing SVCs must still be specified by the solution. Fixed capacitors and SVCs must be specified as described in section 5.2.

To make the problem more realistic, four contingencies are also considered. This system combined with these five cases (the base case and the four contingencies) will be referred to as the *Alpha study*. The test system can be found in appendix B.

To provide a more interesting test, a second test system was developed named the *Beta* study. This study uses a modified version of the Alpha system; the system has been stressed by setting the real power generation of three generators – XNX81, XQX81 and XWX81 – to zero. This simulates the loss of three generators: a significant contingency. Due to the large deficit in real power generation the system will be highly stressed, with the possibility of system voltages being dangerously low. Such a situation would normally require additional reactive power to support the system voltages.

8.2 Experiments using the Beta study: GA parameter tuning

Section 4.6 introduced a number of GA parameters that can effect its performance.

Trials were conducted to explore the effects of varying the parameters of the three GA implementations defined in section 7.8: the SGA, IGA and PGA.

8.2.1 Varying the probability of mutation

For each algorithm, a number of trials were performed using different mutation rates. As discussed, to aid interpretation of the results the mutation rates have been normalised: they are expressed as functions of the string length. Thus, a normalised mutation rate of one means that a mutation rate was selected to provoke at least one mutation per string per generation, while a normalised mutation rate of two causes at least two mutations per string per generation. For reference, the conversions between the normalised mutation rates and the specific mutation rates are given in table 8.2.

At each of the mutation rates, five trials of two hundred generations were carried out, using a population of one hundred. At the end of each trial the GA presented the optimal solutions found. Due to the fact that this is a multi-objective optimisation the GA found more than one optimal solution, but because in these trials the extra objectives were only there to steer the search, most of the dominant solutions could be disregarded. For example, a solution may be presented as an optimal solution because it required no installation, but violates several busbar limits. Such a solution is obviously unacceptable, so the final solution from each trial was chosen according to the following rules, which are evaluated in order.

Table 8.2: Conversion between the normalised mutation rates used and the specific mutation rates using a string length of 500 bits

Mutations per string	Mutations per bit
0.25	0.00050
0.50	0.0010
0.66	0.0013
1.00	0.0020
2.00	0.0040
4.00	0.0080
6.00	0.012
8.33	0.017
12.50	0.025

1. **Security rule:** The solution must not violate any busbar constraint for any contingency. Only solutions that satisfy the "zero busbar violations" constraint may be considered by rule two.

2. **Minimum cost rule:** The chosen solution will have the lowest reactive power cost.

For this problem, by using the above rules only one optimal solution will be found from each trial.

It was found that for each algorithm there was a region of mutation rate values that produce useful results, when the mutation rate moves out of this region, results quickly become extremely poor. A summary of the results for each of the three algorithms, with mutation rates that are in or around this area of productively, is presented in figure 8.1.

The mean result from each set of trials is presented, with the values for each algorithm connected with lines. Each mean value is based on a sample

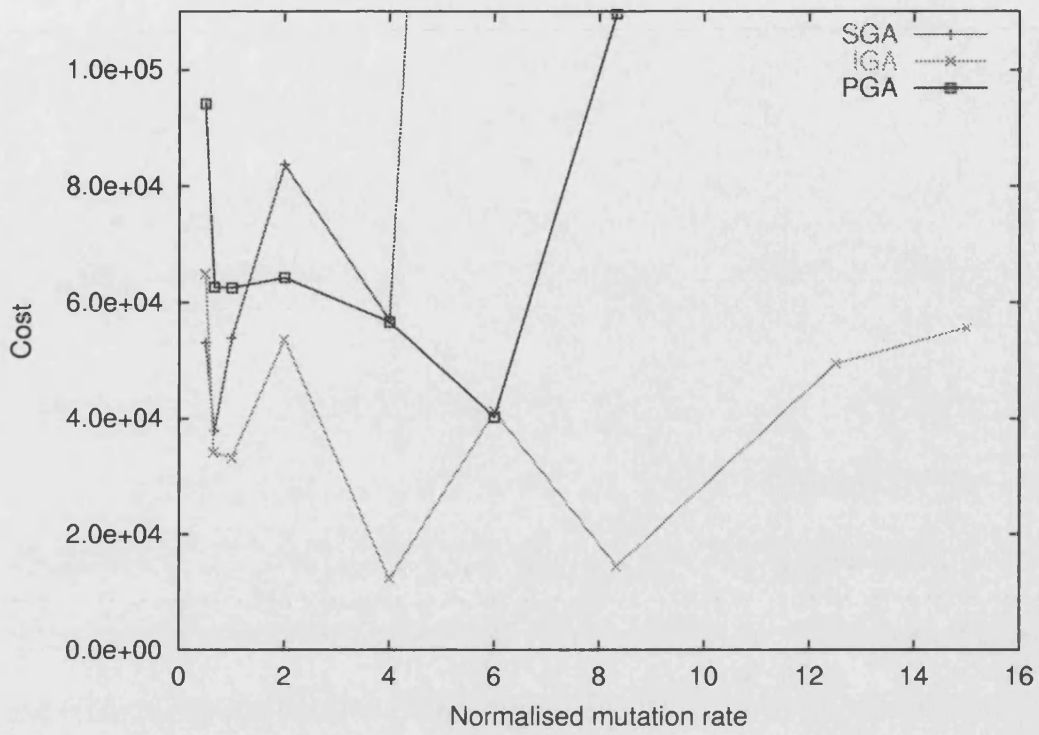

Figure 8.1: Summary of the cost of solutions found by each algorithm, for five trials, at various normalised mutation rates, with reproduction and crossover at 100%. Points show the mean cost of solutions, with lines connecting the points of each algorithm.

size of five trials. With reference to this figure, it can be seen that the IGA generally outperforms the other two algorithms and with a normalised mutation rate of four (that is, four mutation per string per generation) the mean solutions are superior to those generated in any other set of trials. The PGA, discouragingly, appears to be the weakest algorithm, although in its favour, it does demonstrate a greater tolerance than the SGA to the effects of varying the mutation rate; ideally, an algorithm should produce high quality solutions without the requirement for excessive adaptation to the specific problem.

At the extremities of the mutation rates for which results are presented the performance is worst, which is a predictable characteristic. The cases in

which mutation is a rare event – the points to the left of the graph – are poor, due to insufficient diversity being injected into the population, which causes premature convergence on local optima. With high mutation rates the opposite is true: excessive diversity prevents the population from converging on optima and the GA starts to act as a random search. Between these two limits the most productive mutation rates can be found.

As can be seen, all three algorithms appear to have two or more regions of mutation rate settings that favour well. The IGA, in particular, demonstrates this characteristic most clearly by producing excellent results with a normalised mutation rate of four and eight mutation per string, but less encouraging results with six mutations per string. This could well be due to the random nature of the search combined with the small sample size, but figure 8.2 shows that the mean values at these mutation rates have sufficiently low standard deviations to suggest another, unknown, mechanism could be at work.

It is also interesting to note that the IGA performs optimally with a higher mutation rate than the other two algorithms. As discussed, the IGA crossover operator cuts strings between sub-strings and, therefore, does not inject as much new information into the population as the SGA and PGA crossover operators. These results indicates that the IGA uses the higher mutation rate to counteract this effect: increased information generation via mutation compensates for the reduced potential for information generation of its crossover operator.

To further investigate the performance of the algorithms, the stagnation times were examined. To provide comparable information, the stagnation-time data was preprocessed in the following manner: the stagnation times were integrated, so that the total number of previous generations that yielded no improved solutions for each generation was available. Then, the mean curves

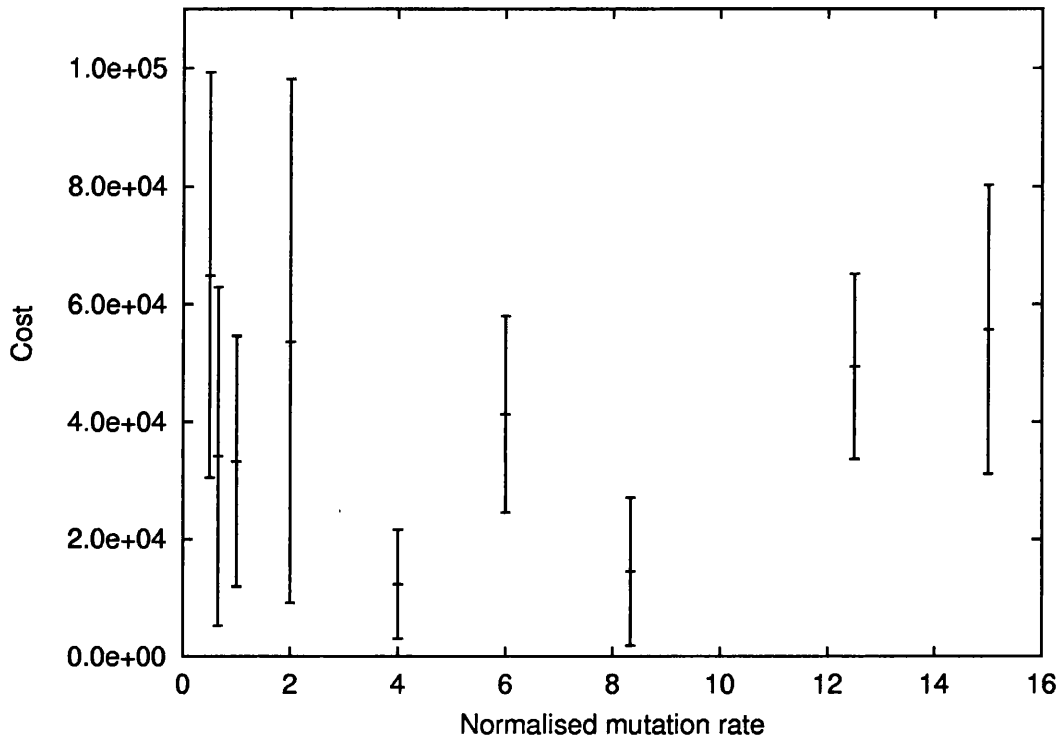

Figure 8.2: Summary of the mean cost of solutions found by the IGA, for five trials, at various normalised mutation rates. Error bars indicate the mean cost +/- one standard deviation.

for each mutation rate of the three algorithm could then be calculated. Figure 8.3 presents the curves for a normalised mutation rate of four: the trials in which the IGA did particularly well.

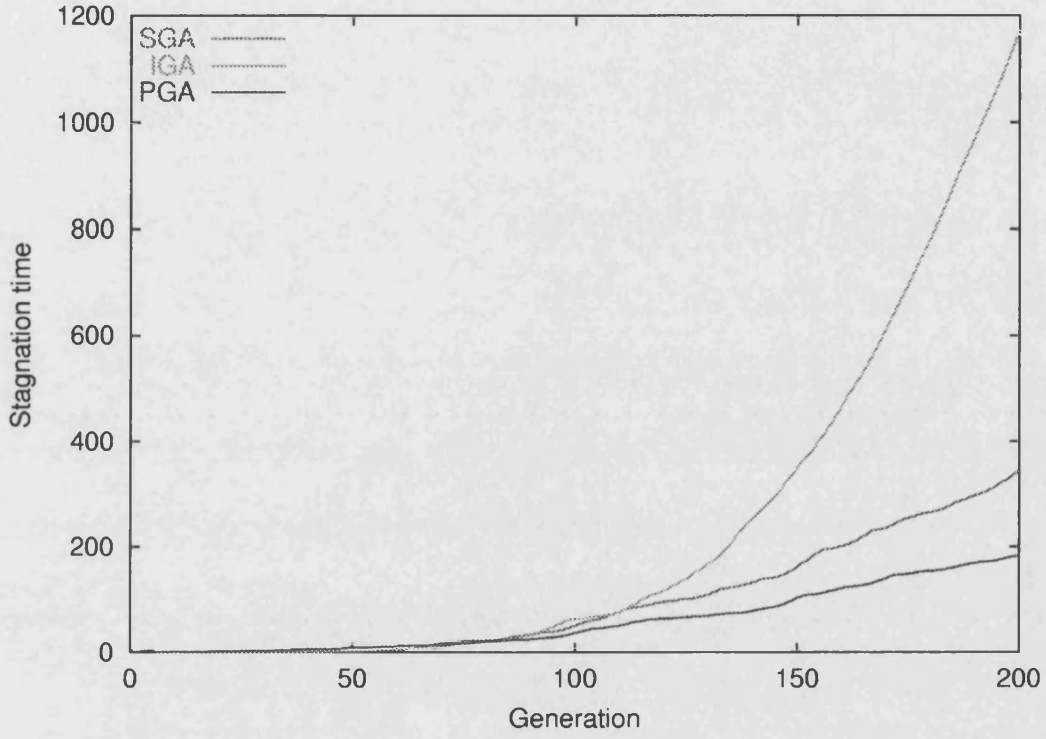

Figure 8.3: Mean integrated stagnation time for the SGA, IGA, and PGA, across four trials, operating with a normalised mutation rate of four mutations per string per generation.

As can be seen, the time spent in a state of stagnation is far greater, on average, for the IGA than for the SGA and PGA trials. This alone is inconclusive, but combined with the knowledge that the resultant solutions found by the IGA were of a higher quality than those of the SGA and PGA trials and convergence tends to slow when GAs approach the global optimum, further strengthens the argument for the superiority of the technique: not only is the IGA finding better solutions, but it is finding them faster. In the case of the other two algorithms, both of which found similar quality solutions, the lower stagnation time of the PGA suggests that it finds improved solutions slower,

but more consistently. It can be argued that, if the algorithms were allowed to carry for more generations, the PGA would be more likely to find improved solutions than the SGA.

To test this hypothesis, a trial was carried out for each algorithm using the best mutation rate for each, as taken from figure 8.2. These were 0.67 for the SGA, four the IGA and six for the PGA. For each trial the algorithms were left for five hundred generations before being stopped. To illustrate the performance of each, figure 8.4 shows the best solution in each generation. Note that each point of the graph represents the best solution in that population at that instant, not the best solution found up to that point.

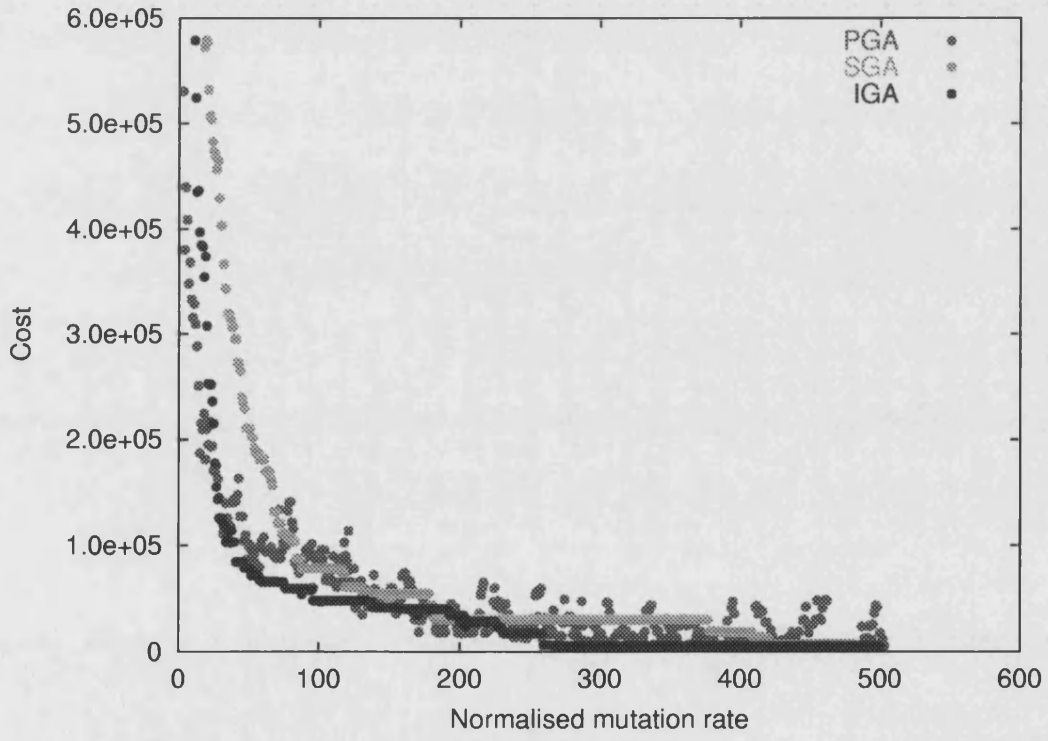

Figure 8.4: Best string in each generation for long trials of each algorithm, each with its optimum mutation rate.

The best solution found in each trial is given in table 8.3.

All three algorithms managed to find solutions that specifying that no compensation is actually required to solve the problem, so the costs specified in

Table 8.3: Best solutions found during 500 generation trials of each algorithm, each with its optimum mutation rate

Algorithm	Cost of best solution found / £	Cost normalised to £4920
SGA	4920	1.00
IGA	4620	0.94
PGA	4710	0.95

table 8.3 are purely functions of the MVAr utilisation. The IGA is clearly the superior to the other two algorithms, as can be seen by the fact that it found the best final solution and its speed of convergence was fastest. By three hundred generations the IGA had already found "no installation" solutions and used the final two hundred generation to further reduce the cost.

The PGA presented a solution that cost five per cent less than that of the SGA, but did not manage the six per cent cost reduction of the IGA. It is interesting to note the very different convergence characteristic of the PGA. From figure 8.4 it can be seen that the SGA and IGA present a smooth curve, indicating that once a good solution is found it tends to stay in the population. The PGA on the other hand shows very erratic convergence. This is because the high mutation rate – on average six mutations per string – was corrupting good solutions before they managed to reproduce sufficiently. The only reason that convergence was maintained is that, as discussed, the population is being monitored and Pareto optimal solutions are being copied and stored; these are then injected back into the population when stagnation is detected.

It appears curious that the IGA does not also display this erratic convergence characteristic: it is at the same mutation rate. The explanation is that the crossover operator of the IGA does not create new information in strings, because it cuts strings *between* integers. The PGA, on the other hand, does not respect these boundaries and, as a consequence, crossover contributes to

new information being generated in the same way as mutation; this explains the high degree of population disturbance demonstrated by the PGA.

Finally, the SGA shows fast initial convergence, and by two hundred generations is matching the PGA. In the later stages of optimisation the speed of improvement falls, and it fails to find solutions of as low a cost as the other two algorithms. This could be attributed to an effect called *premature convergence* in which the population converges rapidly on a local optima, suffering a loss in population diversity, and then being unable to escape from the local optima.

8.2.2 Varying the probability of reproduction and the probability of crossover

The following experiments investigate the effect of lowering the probabilities of reproduction and crossover occurring. Previously, these events occurred with a probability of one: during every generation, each string would be a candidate for selection – and therefore deletion – and each string would be the parent in a crossover operation. In the following test the probability of these event occurring is a half. The population size is maintained at 100.

The operational effect of this change can be visualised as follows: before reproduction takes place, the population is separated into two sub-populations, Y and N. Sub-population Y is a temporary breeding pool and so reproduction is carried out only within sub-population Y. The chance of a string being selected to be in sub-population Y is a half; on average, the size of sub-population Y will be half the size of the complete population. Post reproduction, the sub-populations are recombined. This processes of division-operator-recombination is then repeated for crossover. So each generation, on average, half the population will be involved in the selection tournaments and, independently, half the population will be parents in a crossover operation.

In the following discussions, the notation (r:*reproduction probability*,c:*crossover probability*), will be used to concisely indicate the algorithm and settings under discussion. Therefore, IGA(r:50,c:50) indicates the IGA algorithm operating with a crossover probability of 50 % and a reproduction probability of 50 %.

A summary of the results for each of the three algorithms – the SGA(r:50,c50), IGA(r:50,c50) and PGA(r:50,c50) – using a variety of mutation rates, is presented in figure 8.5. The mean result from each set of trials is presented, with the values for each algorithm connected with lines. Each mean value is based on a sample size of five, 500 generation trials.

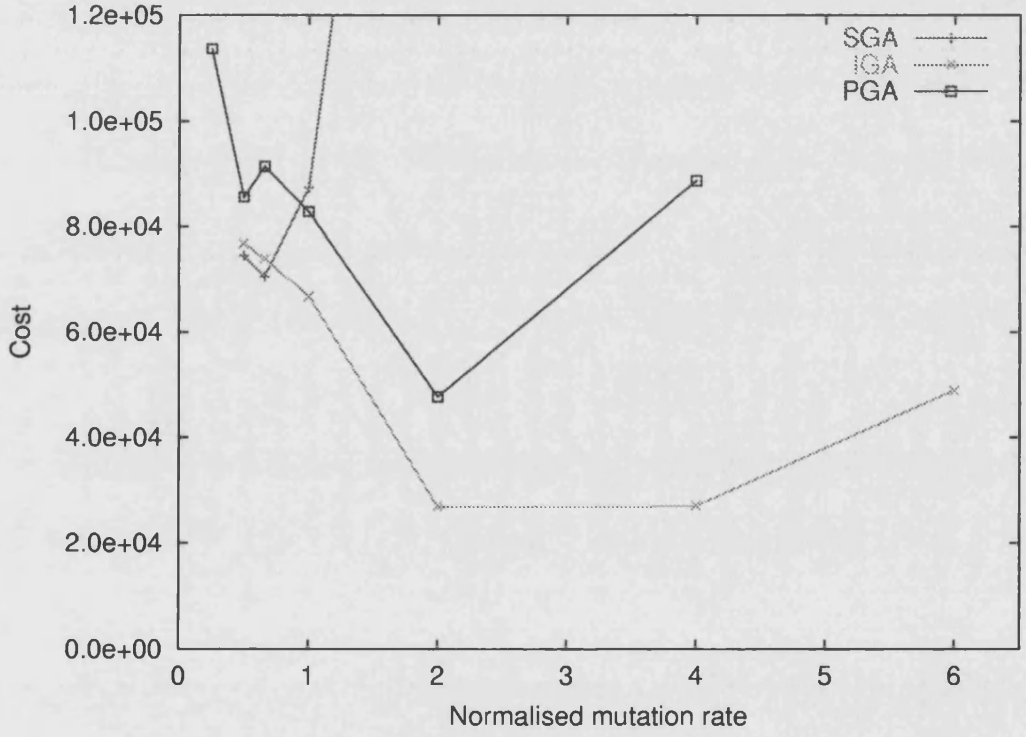

Figure 8.5: Summary of the cost of solutions found by each algorithm, for five trials, at various normalised mutation rates, with reproduction and crossover at 50 %.

None of the algorithms performed any better with the reduced probability of crossover and reproduction, compared with the trials where these occurred with a one hundred per cent certainty. The IGA shows a higher tolerance to

the changing mutation rate, while the SGA and PGA appear far more sensitive to the mutation rate setting which is an undesirable characteristic as it then requires such values to be chosen more carefully for acceptable optimisation. The mutation rates at which the IGA and PGA operate best are noticeably lower than they were with reproduction and crossover at one hundred per cent. This is an expected characteristic as the low reproduction rate means that good solutions are more likely to be destroyed by high mutation rates. What is being seem is, effectively, a "slower" genetic search and no improvement in solution quality has been found within the three hundred generation trials.

The IGA showed the most potential so a five hundred generation trial was executed using the IGA with the optimal mutation rate, according to figure 8.5: two mutations per string. The solution found cost only 5350.78 which is the cheapest solution found so far. To investigate the effects of changing the reproduction rate separately, another two trials were run, using IGA(r:50,c:100) and IGA(r:100,c:50). The complete set of results for the IGA with variable reproduction and crossover rates is given in table 8.4.

Table 8.4: Effect of varying the reproduction rate and crossover rate of the IGA, using mutation rate of two per string, over 500 generations

		Probability of reproduction	
		50	100
Probability of crossover	50	4680	46430
	100	5170	4620

The worst configuration is found with a reduced crossover probability only; the best solution found using this configuration required 420 MVArs to be installed and has suffered a large cost penalty for it. The best configuration is found with a one hundred per cent crossover probability and reproduction probability. The convergence of these four algorithms is given in figure 8.6.

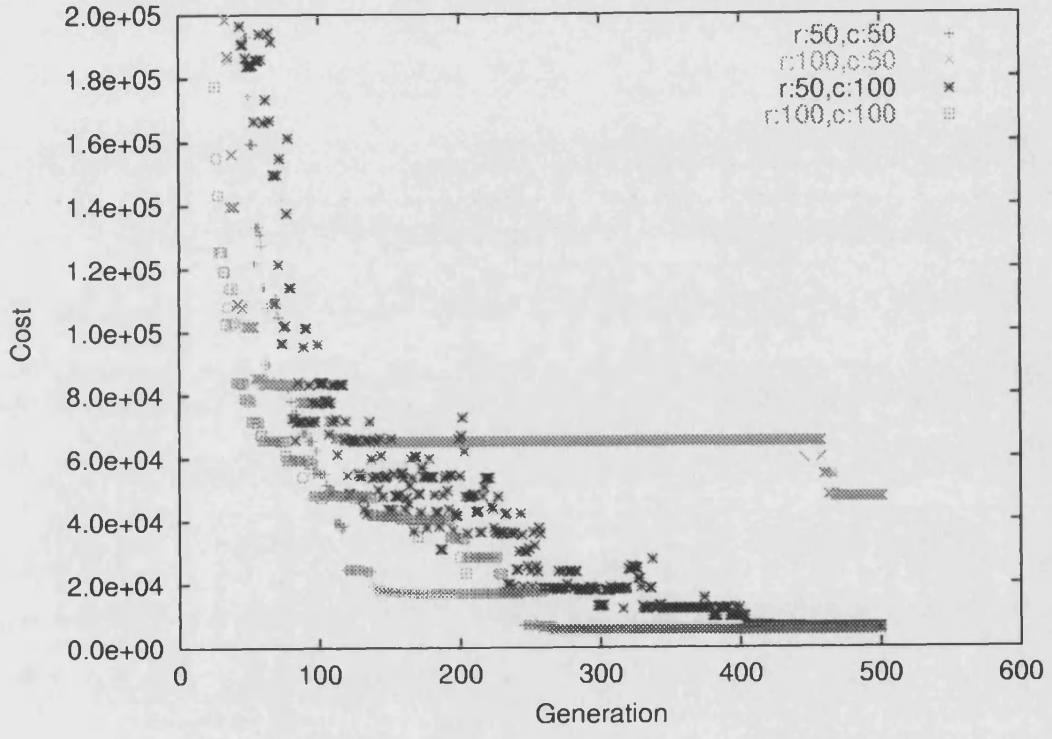

Figure 8.6: Convergence of the IGA using various probabilities of crossover and reproduction.

As can be seen, IGA(r:100,c:50) behaves well over the first one hundred generations, but then improvement drastically slows. This would suggest that the algorithm quickly converged on a local minima, in which it became trapped. This is supported by the parameter configuration: crossover is an exploration operator, with its effect reduced convergence was accelerated, but this lack of exploratory power caused it to get trapped in a local minima.

The opposite is true of the IGA(r:50,c:100). The reduced reproduction resulted in the population converging very erratically, and it can clearly be seen that frequently improved solutions are found and then destroyed by the following generation. It is interesting to note that this algorithm favoured far better, and – probably through good solutions being copied out, and later inserted back into the population – the algorithm still managed to converge on a globally optimal region. When the population had final converged on this optimal region, the lack of exploitation resulting from the low reproduction probably, prevented the algorithm from finding the same quality of solution as the final two algorithms.

The best solutions were found when the probability of reproduction and crossover were balanced. The IGA(r:100,c:100) and IGA(r:50,c:50) both found high performing solutions, the former outperforming the other three. This difference between these two solutions can be attributed to experimental error. This leads to the conclusion that the IGA(r:50,c:50) and IGA(r:100,c:100) are both capable of finding very good solutions.

8.2.3 Varying the population size

Purely considering the best two algorithms found so far, the IGA(r:100,c:100) and IGA(r:50,c:50), the effect of reducing the population size was investigated. Trials were carried out using a population size of fifty: half the previous

population size. The results of these trials and, for compassion, the results using the larger population size are presented in table 8.5.

Table 8.5: Cost of solutions, in pounds, found by the two IGAs using two different population sizes and trial lengths

Number of evaluations	Number of generations	Population size	Algorithm	
			IGA(r:50,c:50)	IGA(r:100,c:100)
25000	250	100	6930	17480
25000	500	50	35910	71700
50000	500	100	4680	4620
50000	1000	50	13880	7820

Obviously, with a smaller population size, optimisation is expected to take a higher number of generations; the important figure is the product of the population size and the number of generations, which gives the number of *evaluations*. Hence, in table 8.5 the results have been grouped into two groups: the first presents results of trials that required twenty thousand evaluations, the second, trials that required fifty thousand evaluations.

As can be seen, the ability of the IGA to find optimal solutions is degraded using the reduced population size and for this problem the "wide and short" approach appears to be preferable.

8.2.4 Conclusion

From these trials, the optimal GA configurations for solving the RCP problem can be defined as follows:

Note the symbolic names: in later sections, these will be used to refer to these two algorithms. It is interesting to note the relationship between the parameters of these two algorithms: the reproduction probability, crossover probability and mutation rate of the second algorithms are half those of the

Table 8.6: Optimal algorithm for solving the RCP problem

Symbolic Name	IGA(r:100,c:100,m:4)	IGA(r:50,c:50,m:2)
Algorithm	IGA	IGA
Population size	100	100
Probability of reproduction	100	50
Probability of crossover	100	50
Mutations per string	4	2
Number of generations	500	500

first. Also of note, these experiments support the observation of Li [1996] that the correct choice of the base algorithm has a more significant impact on performance that the choice of parameters of that algorithm.

8.3 Experiments using the Beta study : comparison between linear programming and genetic algorithms

The evidence required to assess whether a GA based approach can be used to solve the reactive power compensation planning problem can be acquired through comparison with an existing technique that is known to produce quality solutions. Although GAs have been used successfully in the past (as discussed in section 3.2.1) an objective of this work has been to investigate their potential to deal with a combination of practical sized power systems and contingencies. Specifically, their use for optimal planning on the National Grid system. For the purposes of this comparison, the Beta study has been developed, as discussed in section 8.1.

Section 8.3.2 presents the results of compensation planning by the GA based approach developed in this work. Firstly, however, are the results found by the

benchmarking technique, against which GCP will be compared: SCORPION. SCORPION is the reactive power planning tool that is currently being used by National Grid for planning of the England and Wales power system, and therefore provides an excellent, and conclusive, point of comparison. A detailed discussion of SCORPION, and the linear programming optimisation engine it employs, can be found in section 3.5.

8.3.1 SCORPION

The best solution to the Beta problem found by SCORPION is given in tables 8.7, 8.8 and 8.9. As can be seen, in addition to the transformer tap settings and SVC voltage set point settings, the solution requires the installation of 359.69 additional MVArs of reactive power compensation.

Table 8.7: SCORPION solution: generator transformer tap settings

HV busbar	Target voltage	Generator	Tap ratio
XNX4	1.0150	XNX81	1.076799
XQX4	1.0250	XQX81	1.040145
XHX4	1.0250	XHX81	1.063939
XJX4	1.0189	XJX81	1.066436
		XJX82	1.067431
XWX4	1.0161	XWX81	1.075392
		XWX82	1.056957

In order to evaluate the solution, load flow analysis is required. As OPFL02 was developed by National Grid and its results are tried and trusted it was chosen as the mutually acceptable load flow package for solution evaluation and comparison.

To evaluate the SCORPION solution, a study file was created that described the Beta study with the parameter settings and installation requirements

Table 8.8: SCORPION solution: SVC information

Node name	Voltage set-point (p.u.)
XFX4	1.0250
XKX2S	1.0300
XLX2	1.0186
XLX4	1.0250
XZX4	1.0250

Table 8.9: SCORPION solution: installed compensation

Busbar	Size (MVArs) Capacitive
XFX1	325.10
XJX4	20.94
XLX2 (SVC)	13.65
Total	359.69

dictated by this SCORPION solution. This study was then processed using OPFL02, the results of which are given in table 8.10, and the cost breakdown is given in table 8.11.

Table 8.10: Performance of the SCORPION solution

Case	X001 (base case)	X002	X003	X004	X005
Busbar violations	0	79	0	0	0
MW mismatch	0.0370	30.1	0.0581	0.0124	0.0255
MVAr mismatch	0.0155	141	0.0844	0.0104	0.0255
MVAr utilisation	4510	4800	4660	4510	4540
Av. volt. (p.u.)	0.992	0.971	0.987	0.991	0.990
Volt. dev. (p.u.)	0.212	1.318	0.423	0.227	0.300
MW loss	279	342	309	291	292
MVAr loss	3530	4540	3830	3550	3600

Table 8.11: Cost breakdown of the SCORPION solution

	MVArs	Cost per MVAr	Cost
Reactive power utilisation	4510	1	4510
Reactive power installation	360	100	36000
		Total	40510

As can be seen, the solution actually failed to maintain security for one of the contingencies: the load flow failed to find a convergent solution for case X002. The solution violates 79 busbar constraints, which include generation limits, tap setting limits, and so on. Note that SCORPION presented the solution in good faith, but, because it uses an approximation of the load flow calculations, this non-convergence only reveals itself on rigorous examination. This solution therefore is unacceptable because it does not maintain security. Also, from the cost breakdown, it can be seen that the cost incurred by the installations has had a significant impact on the total cost.

Realisable SCORPION solutions

This solution given by SCORPION illustrates one of the drawbacks of an inherently non-discrete optimisation algorithm such as LP: the solution is not actually physically implementable. Firstly, the tap ratios are at an unacceptably high level of precision, and, secondly, the VAr sources are not in commercially available sizes. To make the solution realisable, the tap settings must be rounded to the required precision, and available compensation devices must be chosen, according to table 5.1. Two realisable interpretations of the installation required by the solution found by SCORPION are given in tables 8.12 and 8.13.

Table 8.12: A Realisable SCORPION solution: installed compensation is rounded down to the nearest available size

Busbar	Size (MVArs)	
	Inductive	Capacitive
XFX1	0	300
XJX4	0	0
XLX2 (SVC)	0	0
Sub totals	0	300
Total	300	

Table 8.13: A Realisable SCORPION solution: installed compensation is rounded up to the nearest available size

Busbar	Size (MVArs)	
	Inductive	Capacitive
XFX1	0	360
XJX4	0	60
XLX2 (SVC)	106	150
Sub totals	181	660
Total	841	

To formulate the first siting plan (table 8.12) the VAr installation has been rounded down to the size of the nearest available device, and to formulate the second (table 8.13) the VAr installation has been rounded up. Now the solution requires the installation of either 300 MVArs when rounded down or 841 MVArs when rounded up, compared to the 359.69 MVArs required by the original solution given in table 8.9. The performances of these two realisable solutions were also evaluated using OPFL02: the results are given in tables 8.14 and 8.15.

Table 8.14: Performance of the rounded-down realisable SCORPION solution

Case	X001 (base case)	X002	X003	X004	X005
Busbar violations	0	97	0	0	0
MW mismatch	0.0384	47.8	0.0109	0.0125	0.0257
MVAr mismatch	0.0163	287	0.0524	0.0104	0.0256
MVAr utilisation	4560	5170	4410	4560	4590
Av. volt. (p.u.)	0.992	16.2	0.987	0.991	0.990
Volt. dev. (p.u.)	0.212	2.75	0.430	0.234	0.305
MW loss	280	349	309	290	290
MVAr loss	3550	4690	3850	3560	3610

The rounded-down realisable solution has, unsurprisingly, made the situation worse. Table 8.14 reveals an increased number of busbar-constraint violations for the problematic case X002: the MVAr mismatch, for example, has doubled. This is not an acceptable RCP solution for the Beta study.

Table 8.15 tells a different story: by rounding up the sizes, a solution has been found which satisfies all constraints for all cases. In addition, an improvement in system voltages and a reduction in VAr losses, mismatch and, interestingly, MVAr utilisation can also be seen. This reduction in VAr utilisation could be attributed to the fact that there are now increased sources of VArs in the areas where they are actually needed, and these sources are being used to satisfy

Table 8.15: Performance of the rounded-up realisable SCORPION solution

Case	X001 (base case)	X002	X003	X004	X005
Busbar violations	0	0	0	0	0
MW mismatch	0.0353	0.187	0.0577	0.0123	0.0253
MVAr mismatch	0.0144	0.126	0.0845	0.0104	0.0255
MVAr utilisation	4430	4570	4580	4440	4460
Av. volt. (p.u.)	0.992	0.973	0.987	0.992	0.990
Volt. dev. (p.u.)	0.203	1.130	0.415	0.216	0.295
MW loss	279	340	308	290	291
MVAr loss	3520	4480	3820	3540	3580

demand. Previously, other sources, further from these critical points, where required to supply the demand.

As Henderson *et al* [1958] pointed out, the viability of transmitting VArs is severely degraded by increases in distance, thus explaining the previous increase in VAr utilisation and VAr loss. In fact, from examining the load flow results in detail, it was found that in the previous cases, where case X002 caused significant busbar constraint violations, many of these violations were centred around the three busbars chosen for additional support. For example, the two generators connected to busbar XJX4 (XJX81 and XJX82) were both operating outside their reactive limits and the three lowest system voltages could be found at the busbar XLX2 and the direct neighbours of XLX2 and XFX1. With extra support of the rounded-up realisable solution, the voltages had been brought within acceptable limits and the SVC at XLX2 was operating close to it maximum capacitive generation. The difference in busbar voltages between the original solution and the improved solution are shown in figure 8.7.

As can be seen, the entire system voltage profile has been improved. As the rounded-up realisable solution is the only one of the three SCORPION

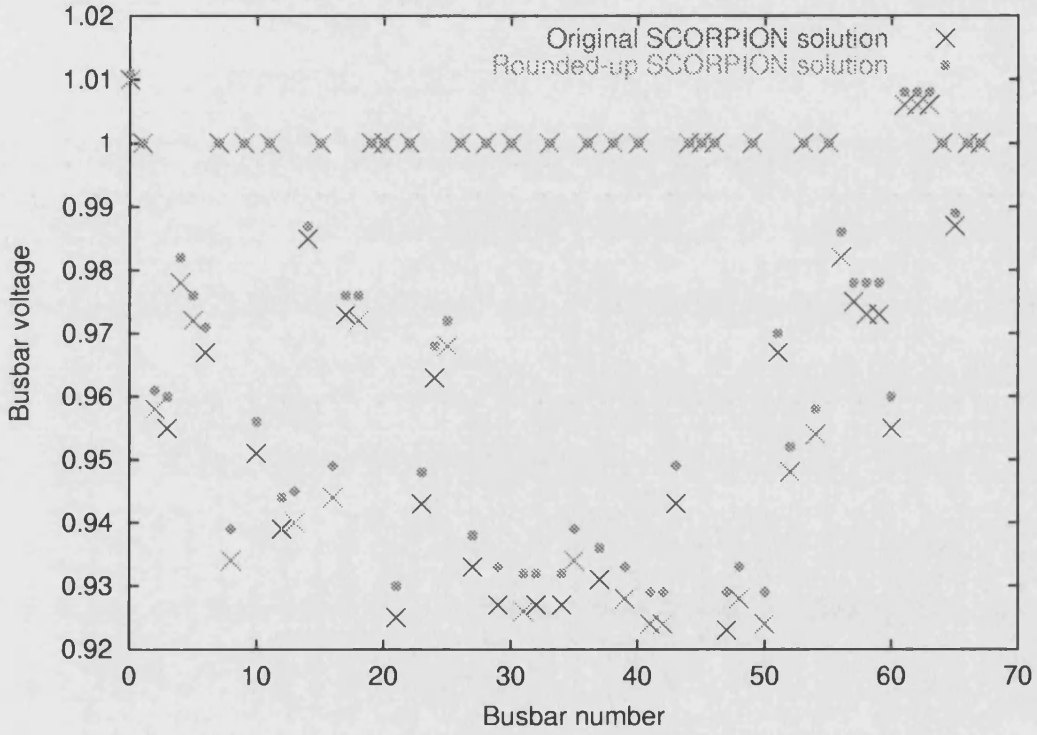

Figure 8.7: Comparison of the voltage profiles using the original SCORPION solution and the rounded-up solution for case X002

solutions discussed that is fully secure, it will be used as the benchmark against which GCP will be compared. Table 8.16 presents the cost analysis of this solution.

Table 8.16: Cost breakdown of the rounded-up realisable SCORPION solution

	MVArs	Cost per MVAr	Cost
Reactive power utilisation	4430	1	4430
Reactive power installation	841	100	84100
		Total	88530

8.3.2 GCP

Section 8.2, presented the results of experiments conducted to find an optimal formulation of the GA used by GCP. Section 8.2.4 conclusions from these experiments and presented the two optimal GA formulations that achieved the highest performance. Of these two algorithms, the first algorithm, labelled IGA(r:100,c:100,m:4), will be used in the following comparison.

Figure 8.8 illustrates the convergence of the population. The very expensive solutions have not been shown: of each population, only solutions with a cost of less than 10^6 pounds have been shown. The extremely expensive solutions are those that incur great costs due to producing non-convergent load flows. The high number of such solutions is an indication of the large portion of the problem space that produces infeasible solutions.

To better understand how the algorithm converges on the optima, figure 8.9 shows only the best string present in each generation, truncated to one hundred thousand pounds. Also shown is the stagnation time: the number of generations that had passed since a solution had been found that was better than any other previously found.

As can be seen, improvement is made consistently over the first seventy

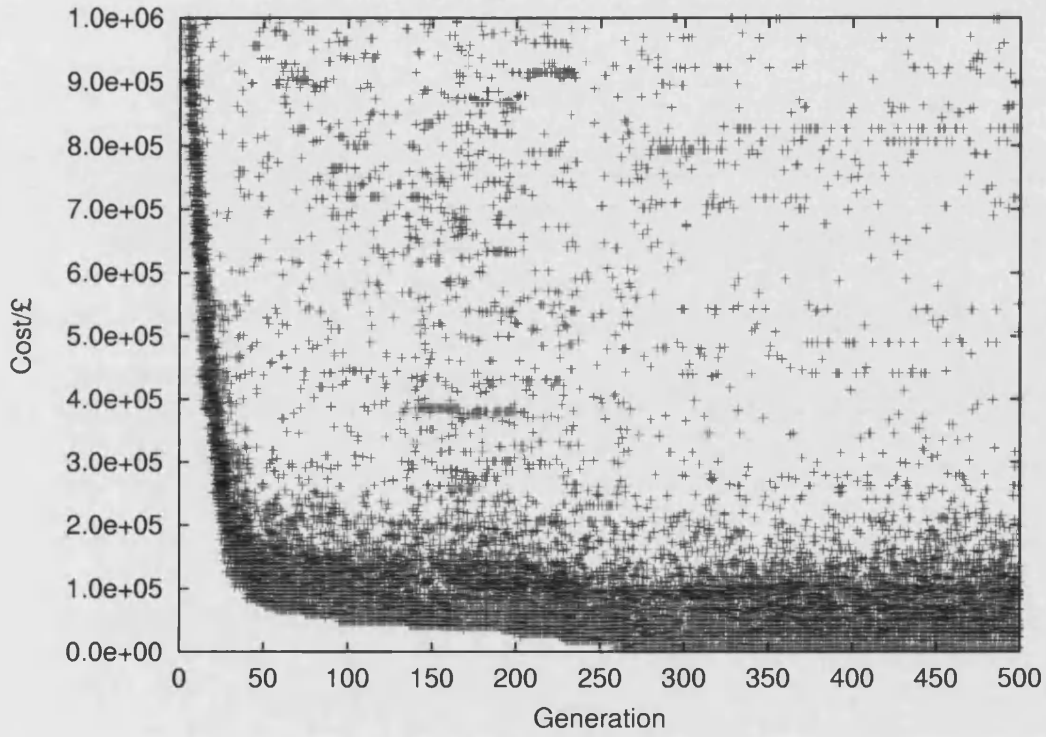

Figure 8.8: Cost of the solutions contained in each population against generation

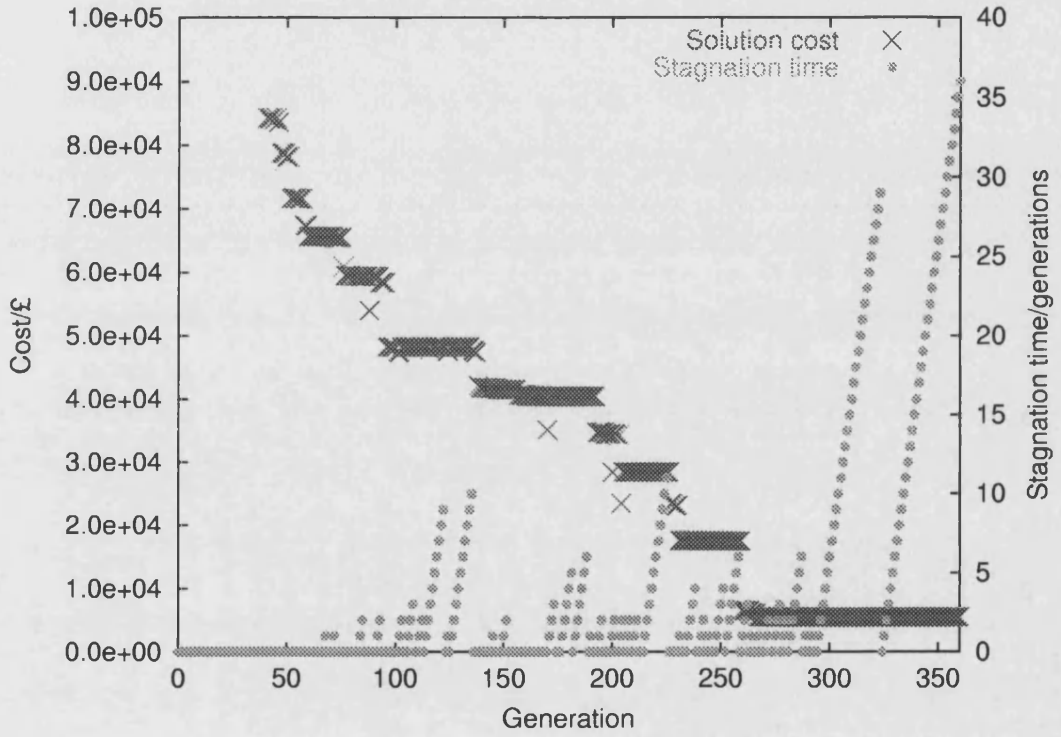

Figure 8.9: Cost of best solution in each population and population stagnation time against generation

generations. At 124 generations into the optimisation, improvement gets significantly interrupted and no improved solutions are found for the next ten generations. At this point the anti-stagnation mechanism is triggered and the set of the best strings found up to that point are inserted back into the population. It can be clearly seen that this successfully prevented further stagnation and significant improvement was found in the next population. At several other points in the trial a similar occurrence can be seen, each time optimisation is successfully encouraged. By generation 295 improvement has all but stopped, and from 325 generations on no improvement is found for the remainder of the five hundred generation trial.

The details of best solution found in the trial – which requires no VAr installation – is presented in tables 8.17 to 8.19:

Table 8.17: GCP solution: generator transformer tap settings

HV busbar	Target voltage	Generator	Tap ratio
XNX4	1.05	XNX81	1.134
XQX4	1.08	XQX81	1.158
XHX4	1.1	XHX81	1.168
XJX4	0.98	XJX81	1.034
		XJX82	1.033
XWX4	1.01	XWX81	1.085
		XWX82	1.067

From the installation and performance data, the cost of the solution can be calculated. The formulation of the reactive power cost is given in table 8.20.

As can be seen, a solution has been found that relies entirely on the optimal setting of the preventive taps, and no installation is required. This has resulted in the solution being orders of magnitude cheaper than the SCORPION solution.

Table 8.18: GCP solution: SVC information

Node name	Voltage set-point (p.u.)
XFX4	1.09
XKX2S	1.09
XLX2	1.06
XLX4	1.07
XZX4	1.06

Table 8.19: Performance of the GCP solution

Case	X001 (base case)	X002	X003	X004	X005
Busbar violations	0	0	0	0	0
MW mismatch	0.0191	0.0538	0.0807	0.0237	0.0154
MVAr mismatch	0.0103	0.0232	0.0132	0.0337	0.00223
MVAr utilisation	4630	4720	4590	4470	4640
Av. volt. (p.u.)	1.00	0.994	0.997	1.00	1.00
Volt. dev. (p.u.)	0.329	0.551	0.393	0.337	0.372
MW loss	290	344	324	306	307
MVAr loss	3760	4510	4090	3790	3840

Table 8.20: Cost breakdown of the GCP solution

	MVArs	Cost per MVAr	Cost
Reactive power utilisation	4630	1	4630
Reactive power installation	0	100	0
		Total	4630

8.4 Cost reduction and voltage deviation reduction: a multi-objective optimisation

Solutions generated using the previous objective function, in which cost was optimised, display an interesting and possibly undesirable characteristic. As the system constraints are expressed through the "busbar violations" measure, unless a solution violates a limit, it is deemed acceptable. The concern is that the GA, in its search for minimum cost solutions, can tend to find solutions that stress the system toward its limits. This was particularly noticeable in respect to the voltage profile. This, of course, is quite acceptable under the constraints of the problem. Each busbar has a limit on the acceptable voltage at that point and as long as the voltage is within these limits the solution is valid. The problem is that the solutions found by the GA, especially in the cases where extremely cheap solutions were found, pushed the voltages at many of the system busbars to their limit; for example the cheapest solution (£4630; table 8.20) found by IGA is responsible for a voltage deviation of 1.89 (table 8.19). This cumulative voltage deviation may well be unacceptable, because if the system is highly stressed for the base case it would be less likely to maintain security in some other unconsidered contingency.

One way to correct this problem, apart from optimising for every possible contingency, is by including the minimisation of total voltage deviation – summed across all cases – as an objective of optimisation. This would then make the objective function truly multi-objective with the complementary objectives of voltage deviation minimisation and cost minimisation. As discussed, the predicted effect of this is that the algorithm finds more than one optimal solution, none of which are dominated by any other solution.

Using the "reactive power cost minimisation and optimal voltage profile" objective function, as defined in section 6.4.2, and the IGA(r:100,c:100,m:4),

as defined in table 8.6, a five hundred generation trial was carried out and a set of Pareto-optimal solution were found. These are given in figure 8.10.

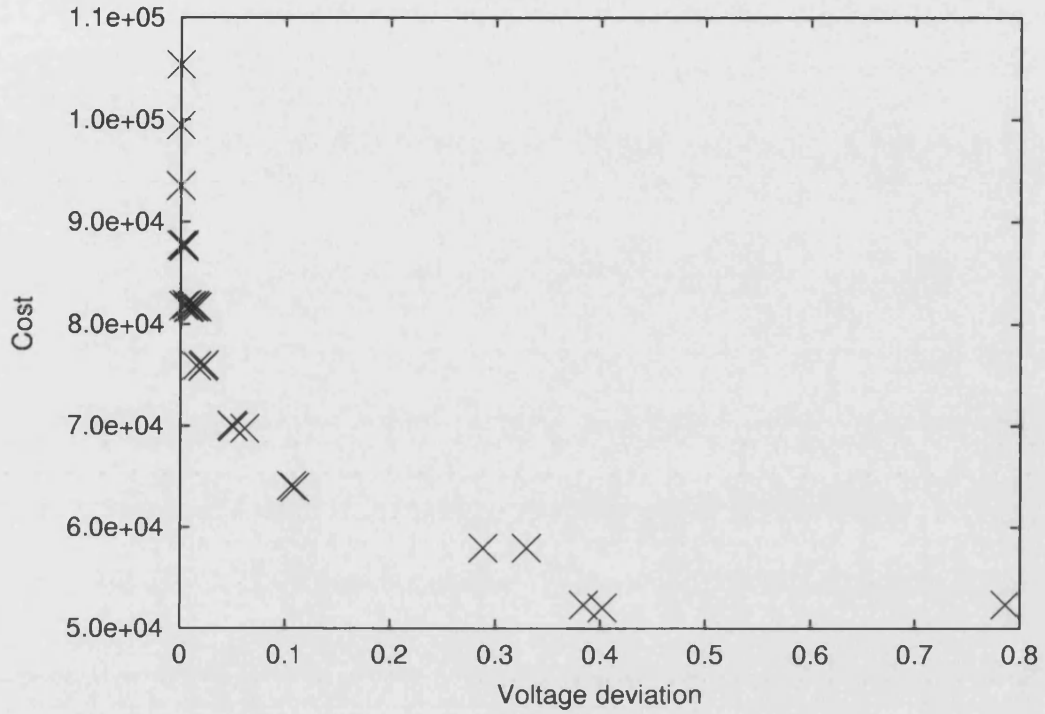

Figure 8.10: Pareto optimal solutions found by the IGA(r:100,c:100,m:4) with a 500 generation trial

In this case, the concept of Pareto dominance states that for one solution to dominate another it must lie in the area to the left and below the dominated point, or on the perimeter of this region. As can be seen, none of the points in the plot are dominated by any other. The curve that these points lie on is called the *Pareto front*.

As expected, there are a number of optimal solutions, one of which would need to be selected to be implemented. This could be based on some idea

of best compromise or, alternatively, all the solutions could be passed on for further evaluation, perhaps to asses their effects on system dynamics or their environmental impact.

From figure 8.10 it can be seen a certain granularity, or grouping of solutions. This is down to the minimum achievable change in installation cost. As discussed, the smallest change in installation is attributed to fixed compensators: sixty MVAr capacitors to be precise. At a cost of one hundred pounds per MVAr, the smallest change in installation cost is six thousand pounds. This is responsible for the observed jumps in cost. The smaller changes, the difference between very similar solutions, is due to different utilisation costs.

To investigate the effect on the voltage profile of this multi-objective optimisation, figure 8.11 presents a comparison of the voltage profiles for case 2 of the study. The figure shows the profile generated by this multi-objective solution, the profile generated by the solution found by the single objective optimisation – presented in section 8.3.2 – and the rounded-up SCORPION solution – presented in section 8.3.1.

From the figure it can be seen that, when compared to the single-objective GCP and SCORPION solutions, the multi-objective GCP solution is superior as a much improved voltage profile is observed: almost all busbar voltages are closer to the ideal and, in particular, the outlying voltages have been significantly corrected.

Observe that, with the extra objective of minimising voltage deviation all of the solutions found are at least an order of magnitude more expensive than the solution found in the single objective case. This highlights a drawback of this multi-objective formulation, because it is known that there exist solutions far better than those found in terms of one objective. As quoted above, the IGA solution that incurred a cost of £4630, with a voltage deviation of 1.98, is far cheaper than any solution found in this multi-objective formulation, but it is of note that this solution lies at the extremities of the multi-objective

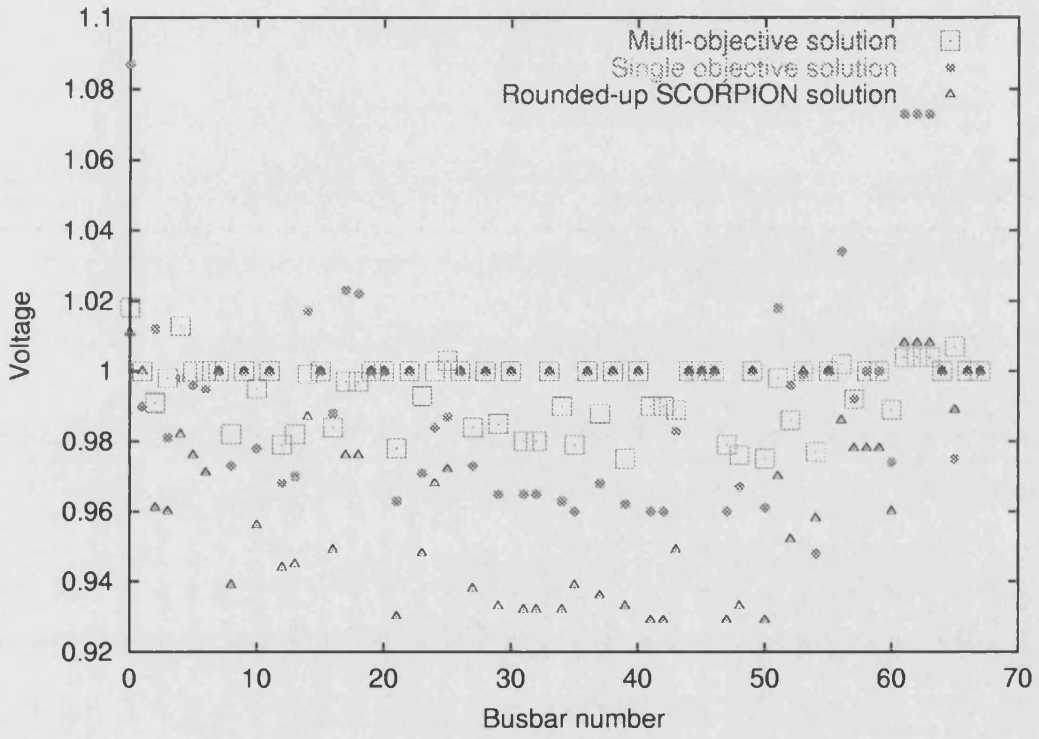

Figure 8.11: Comparison of the voltage profiles, for case 2 of the Beta study, between the solution found in the single objective search, the multi-objective search, and the rounded up SCORPION solution

Pareto front. In conclusion, care must be taken to only use a multi-objective formulation when really required, because the algorithm will tend to discover the compromising solutions towards the front of the Pareto front.

8.4.1 Comparison with SCORPION

As this multi-objective formulation is still dealing with the same problem (the Beta study) comparison with SCORPION is still valid. With reference to table 8.16 (the rounded-up SCORPION solution) the cost of the solution was £88530. With reference to table 8.15, the total voltage deviation of this solution can also be calculated:

$$\begin{aligned}\text{Total voltage deviation} &= 0.203 + 1.130 + 0.415 + 0.216 + 0.295 \\ &= 2.259\end{aligned}$$

So, solutions that are cheaper than £88530, with a voltage deviation of less than 2.259 will dominate the SCORPION solution in terms of these objectives. From figure 8.10, which shows the set of Pareto-optimal solutions, it can be seen that there are 24 solutions that dominate the SCORPION solution. These solutions are given in table 8.21, sorted in terms of cost.

For further analysis, one of the middle solutions – number twelve, in fact – from the table has been selected. The full specification of the solution can be found in the following tables: 8.22,8.23 and 8.24.

It can be seen from table 8.24 that, like SCORPION, compensation has also been placed at busbar XFX1. Unlike SCORPION, the IGA solution comprises purely of fixed capacitors and has placed four smaller devices, rather than the three of SCORPION's solution.

Tables 8.25 and 8.26 presents the performance of this solution.

Table 8.21: Solutions to the multi-objective problem that dominate the SCORPION solution

Number	Cost / £	Total voltage deviation
1	87720	0.0031
2	87690	0.0010
3	87540	0.0031
4	81910	0.0061
5	81810	0.0133
6	81810	0.0031
7	81800	0.0103
8	81660	0.0154
9	81650	0.0123
10	81620	0.0082
11	81610	0.0072
12	75970	0.0225
13	75890	0.0154
14	75820	0.0215
15	70030	0.0492
16	69880	0.0503
17	69740	0.0616
18	64080	0.1060
19	63930	0.1091
20	57970	0.3295
21	57960	0.2883
22	52360	0.7865
23	52340	0.3842
24	52050	0.4022

Table 8.22: A typical GCP multi-objective solution: generator transformer tap settings

HV busbar	Target voltage	Generator	Tap ratio
XNX4	1.03	XNX81	1.099
XQX4	1.01	XQX81	1.005
XHX4	1.02	XHX81	1.052
XJX4	1.01	XJX81	1.070
		XJX82	1.069
XWX4	1.02	XWX81	1.096
		XWX82	1.078

Table 8.23: A typical GCP multi-objective solution: SVC information

Node name	Voltage set-point (p.u.)
XFX4	1.06
XKX2S	1.02
XLX2	1.04
XLX4	1.05
XZX4	1.05

Table 8.24: A typical GCP multi-objective solution: installed compensation

Busbar	Size (MVArs) Capacitive
XFX1	120
XPX2	60
XUX2	240
XGX2	300
Total	720

Table 8.25: Performance of the typical GCP solution

Case	X001 (base case)	X002	X003	X004	X005
Busbar violations	0	0	0	0	0
MW mismatch	0.00	0.0336	$4.19e^{-3}$	$7.80e^{-3}$	0.0104
MVAr mismatch	0.00	0.0103	0.0253	0.0233	0.0320
MVAr utilisation	3970	400	3960	3980	4000
Av. volt. (p.u.)	1.00	0.995	0.999	1.00	1.00
Volt. dev. (p.u.)	0.00	0.0225	0.00	0.00	0.00
MW loss	273	319	302	284	285
MVAr loss	3410	4080	3690	3430	3460

Table 8.26: Cost breakdown of the GCP solution

	MVArs	Cost per MVAr	Cost
Reactive power utilisation	3970	1	3970
Reactive power installation	720	100	72000
		Total	75970

Comparison with the SCORPION solution's performance, presented in table 8.15, it can be seen that there is significant improvement across the board: a reduction in the complex power mismatch and loss, MVAr utilisation and voltage deviation has been achieved. This has been achieved at a lower cost and requires a reduced amount of compensation to be installed.

8.5 Execution time data

GCP is experimental software and was developed with functionality, reliability and ease of modification in mind. Although fast execution time was not a prime concern, data pertaining to the execution time may still be of value.

The time taken by GCP, for a single optimisation run, is a function of the problem size (the network size and the number of cases concidered), the population size, the number of generations, and the frequencies of the genetic operators occuring (for example, the mutation rate).

The problem size affects both the speed at which the data can be manipulated within memory and, more significantly, the time taken by the load flow calculations. Due to the fact that an external load flow program has been used, each evaluation of a candidate solution requires file creation, file writing, file reading and file parsing in addition to actually solving the load flow equations themselves. This makes it a time consuming activity, and does much to dictate the overall running time.

Also, the genetic operators effect the execution speed: higher mutation rates, probabilities of crossover and probabilities of reproduction all have an effect on run time. Also of note is the fact that each time a string is changed in any way, it requires reevaluation. If a string is not affected by mutation and is not involved in crossover, it may retain previously evaluated performance data.

Finally the major contributer to run time is the product of the population size

and the number of generations. This indicates the number of candidate solution evaluations required, each requiring load flow evaluation. The number of evaluations is also proportional to the number of mutation/reproduction/crossover events that take place during optimisation.

A 200 generation optimisation using a population size of 100 strings and a low mutation rate (1 mutation every 4th string) takes 1 hour 45 minuets. A 500 generation optimisation using a population of 100 strings and a high mutation rate (4 mutations per string), 5 hours 50 minuets to complete.

8.6 Conclusions

In the previous experiments, three solutions present themselves for comparison: the feasible, rounded-up SCORPION solution (SCORPION), the solution found by GCP using the cost-reduction objective (GCP-SO) and the solution selected from the Pareto set of solutions found by GCP using the cost reduction and voltage deviation reduction objective set (GCP-MO). The performance of these solutions are summarised in table 8.27.

In the table "total" implies that the associated measure has been summed across the five cases and the "average voltage" is the mean average voltage of the five cases. To further aid comparison, table 8.28 presents a number of these measures in a normalised form. In each case the values have been normalised against the values associated with SCORPION – thus all SCORPION values are unity. Only the measures that can meaningfully be normalised have been shown.

As can be seen, both GCP solutions have found feasible solutions at a lower cost than SCORPION. The GCP-SO achieved the greatest cost reduction – 5.2% of the cost of the SCORPION solution – while reducing voltage deviation by 12.4% and MVAr mismatch by 68.4%, and increasing losses and utilisation.

Table 8.27: Summary of the performance of the rounded up SCORPION solution the single-objective GCP solution and the selected multiple-objective GCP solution

Case	SCORPION	GCP-SO	GCP-MO
Total Busbar violations	0	0	0
Total MW mismatch	0.318	0.1927	0.0440
Total MVAr mismatch	0.261	0.08263	0.0909
Total MVAr utilisation	22480	23050	16310
Av. volt. (p.u.)	0.986	0.998	0.998
Total volt. dev. (p.u.)	2.26	1.982	0.0225
Total MW loss	1510	1571	1463
Total MVAr loss	18900	19990	18070
Cost	88500	4630	75970

Table 8.28: Summary of the performance of the rounded up SCORPION solution the single-objective GCP solution and the selected multiple-objective GCP solution. Values have been normalised against the SCORPION solution.

Case	SCORPION	GCP-SO	GCP-MO
Total MW mismatch	1	0.605	0.138
Total MVAr mismatch	1	0.316	0.348
Total MVAr utilisation	1	1.025	0.725
Total Voltage deviation	1	0.876	0.009
Total MW loss	1	1.040	0.968
Total MVAr loss	1	1.057	0.956
Cost	1	0.052	0.858

The GCP-MO solution has caused improvement in each category. While only reducing cost by 14.2%, the voltage deviation has been reduced by 99.1% and the MVAr mismatch by 65.2%. Also, slight reductions in losses and MVAr utilisation can be seen.

Chapter Nine

Conclusions

The operators of bulk electrical power transmission systems are responsible for the quality and security of the power they supply to the consumer. Ensuring that the voltage at each system busbar remains within its acceptable limit is an essential aspect of maintaining quality and security. Reactive power compensation devices provide an opportunity for controlling voltage and, therefore, ensuring that there is sufficient reactive power compensation is paramount in guaranteeing a secure, high quality electrical power supply.

The objective of the *reactive power compensation planning* (RCP) problem is to define the specification for the installation of new reactive sources which achieves the maximum benefit at the least possible cost, while satisfying system and operational constraints.

To fully evaluate the performance of a possible siting scheme, the operational problem must also be solved. This sub-problem of the RCP problem states that before a system can be evaluated, the parameters of certain devices that exist on the power system must be configured in such a way that they provoke optimal operation of the power system. These devices are the preventive transformer taps and any dynamic compensation devices. These devices require this attention because they also affect the system voltage profile and, therefore, have an influence on quality and security.

Analysis of the literature revealed that proposed solutions to the RCP problem exhibit various levels of complexity. The simplest of these involve only the placement of fixed compensation devices, such as the fixed capacitor, for single system states. This work has been specifically biased towards the planning problem as faced by the maintainers of real-world transmission systems.

For the maintainers of power systems the problem is far more complex and requires the optimisation of many more variables across many system states (intact and contingent). A solution to the complete RCP problem, as dealt with in this work, must address the following issues:

- Both fixed compensation devices and dynamic compensation devices are available for siting.
- Both newly installed and any existing dynamic compensation devices require the voltage set point of their controller characteristic be optimised.
- Preventive transformer taps are optimised through either the optimisation of their tap setting or the optimisation of the target voltage of the busbar connected to the high voltage side of the transformer.
- Security must be maintained during contingent system states.
- There must be no limit, imposed by the software, on the size of the problem: the test system size, the number of contingencies and the number of busbars that may be candidates for installation of compensation must all be unconstrained.

This work presents an investigation into the use of *Genetic Algorithms* (GAs) for practical RCP. The objective has been to asses the performance of such a technique when applied to the planning of practical sized power systems and, in particular, for planning of the England and Wales transmission system, as owned and operated by National Grid.

GAs, the search techniques inspired by natural genetics and Darwinian evolution, have received considerable attention in the past and have been found to be particularly suitable when applied to this type of problem, due to their inherent ability to optimise discrete, multi-model, multi-objective, non differentiable problems. Previous works have shown GA approaches to perform

well, but their application has been limited to either the planning of small test systems or the planning of intact systems with no regard for contingencies. Also, the use of GAs for the optimisation of preventive transformer taps, and the placement and optimisation of dynamic compensation devices has not been fully validated.

This research has resulted in the implementation of a computer package, called *Genetic Compensation Planning* (GCP), using C++ and an object oriented design paradigm.

A formulation of the RCP problem has been defined, based on the planning software called SCORPION, used by National Grid for the planning of the England and Wales transmission system. The formulations is characterised as follows:

- Reactive power cost minimisation. A charge is levied for the installation and use of reactive sources. Optimisation acts to minimise this cost.
- Optimal base, secure contingencies. Only the cost of the base case is minimised. Solutions need only guarantee that the system maintains security during contingent conditions.

A test problem was developed with the cooperation of National Grid, called the *Beta study*, for validation of the algorithm. The Beta study is a 67 busbar system with five different system states (the base case and four contingencies), which has been highly stressed via the removal of three generators.

Three GA based algorithms were implemented: a *Simple Genetic Algorithm* (SGA) as defined by Goldberg [1989], an *Integer coded Genetic Algorithm* (IGA), and a final algorithm that drew inspiration from studies of the role and function of proteins which was named the *Protein coded Genetic Algorithm* (PGA). The IGA and PGA both use a string of integers to represent candidate solutions, with each integer mapping to one parameter of the solution. In

addition, they both utilise the idea of parameter templates. This is a string of integers, of the same form as candidate solutions, but with each integer encoding the number of possible values that the associated parameter may take. The parameter template is enumerated before the GA is initialised and can then be used during initialisation of candidate strings and during genetic operations. In both cases the parameter template is used as a reference to prevent illegal parameter values from occurring in candidate strings. This circumvents many of the problems that afflict GAs that utilise a binary coded string representation by preventing problem space distortion.

Preliminary experiments indicated that the IGA outperformed the other two algorithms. The IGA was further investigated and two sets of parameters were enumerated that incited optimal performance from the algorithm. One of these two algorithms was then designated the algorithm that GCP would use in further studies. The chosen algorithm was the IGA using a population of one hundred strings, optimising over five hundred generations with crossover and reproduction occurring with a one hundred percent certainty and a mutation rate which provoked the occurrence of at least four mutation occurring per string.

Following on from this, the performance of the GCP was compared to that of SCORPION, which uses a linear programming and back-tracking technique for optimisation. GCP was shown to find solutions that were orders of magnitude cheaper than those found by SCORPION. A 94.8 % reduction in cost was shown while reducing voltage deviation by 12.4% and MVAr mismatch by 68.4%

GCP was further improved by implementing multi-objective optimisation: minimising both the cost of solutions and, to further ensure security, the voltage deviation of the base case and of the contingent systems. It was found that of the set Pareto optimal solutions presented, solutions were found that were cheaper than solutions found by SCORPION and performed extremely

well: an improvement in the voltage profiles, a decrease in complex power mismatches, and a reduction in MVAr utilisation was observed. One of these solutions was selected for analysis as the "best compromise" solution. Comparison against SCORPION revealed a reduction in cost of 14.2%, voltage deviation reduced by 99.1% and the MVAr mismatch reduced by 65.2%. Also, slight reductions in losses and MVAr utilisation were achieved, and the average system voltage was found to be closer to the ideal.

In conclusion, GAs have been shown to perform very well when applied to a full, practical sized, multi-contingent RCP problem and can find solutions that outperform a "tried and trusted" solution. Using a single cost-reduction objective yields the cheapest solutions, but multi-object optimisation may be advisable in the interests of improved quality and security. It is therefore recommended that genetic algorithms be considered an appropriate and powerful algorithm for reactive power compensation planning: siting of both static and dynamic compensation devices and the optimisation of preventive transformer taps for large, multi-contingent highly stressed systems.

Chapter Ten

Future Work

This chapter presents a number of opportunities for further investigation that follow on from the work presented herein.

10.1 Third generation device placement

As discussed in section 5.2.3, although STATCOMs could be placed, productively, in the locations chosen for SVCs, this may not be optimal as it neglects the ability of such devices to exchange real power with the system (see section 2.3.3). One problem that would have to be overcome is highlighted by the use of the expression *exchange power* as opposed to *generate power*. As discussed, the STATCOM can operate as a AC/DC converter, and store the DC energy in a capacitor. This means that the STATCOMs ability to generate (or absorb) power is finite: it can only operate in this mode until the capacitor is fully charged (or fully drained). This operation is, therefore, not compatible with steady state modelling, as has been utilised by GCP for solution evaluation; dynamic simulation would be required to fully model a STATCOM operating in this manner.

10.2 Operational SVC placement

One standard aspect of the reactive power compensation problem is that it is a *planning* problem – in which the planning horizon is measured in years – quite separate to the operation problem (see section 2.2.1).

With the invention of relocatable SVCs, for which the point of connection on the transmission system may be changed in a period of months [Luckett, 1999], the distinction between planning and operation becomes less clear. This has a number of connotations for the placement of devices. For example, planning software that uses information about the relocation time of devices could be advantageous: software that, in fact, optimises the placement and *scheduling* of relocatable SVC devices.

Another consideration that could affect the shorter planning horizon, as provided by relocatable devices, is that there will be less uncertainty in the predicted system state. To date, this uncertainty has two effects on planning. Firstly, in the interests of security, the planned system may have to conform to higher levels of redundancy than may otherwise be required. Secondly, due to the large margins of error in the predicted system state, steady state simulation is considered sufficient for solution evaluation.

The increased accuracy in which the short term system state may be specified suggests that the use of dynamic simulation in compensation planning may be desirable. When dynamic simulation is being used to evaluate candidate solutions, objectives such as stability improvement may be considered [Karlsson *et al*, 1998; DeMarco and Overbye, 1990; Aboreshaid and Billinton, 1999].

To date, compensation placement techniques which consider stability enhancement have tended to exploit purely analytical approaches [Parker *et al*, 1996; Cañizares and Faur, 1999], with objectives concerning security enhancement but not cost reduction.

10.3 Optimisation of device type

Another productive extension to the solution presented herein would be by optimisation of the device type. Although both fixed compensation and

dynamic compensation are both supported by GCP, the sets of candidate busbars for these devices are mutually exclusive; the matter of the kind of device that may be placed at a location is a matter of problem specification, not optimisation. As there is not only a difference in functionality, but also in cost, between fixed compensation and dynamic compensation, device type would also appear to be a candidate for optimisation.

10.4 Problem space reduction

Problem space reduction refers to the act of analysing the problem space, before optimisation has been initiated, with the intention of selecting a subset of that space in which to confine the search. As each point in the problem space represents a candidate solution, reducing the size of this space means that the number of candidate solutions is reduced. This, in theory, implies that the optimisation technique, whatever it may be, will perform better.

One such technique, that has been applied to the RCP problem is weak busbar analysis [Chen *et al*, 1995]. As reactive power compensation is used to correct the voltage profile of a power system, the technique suggests that any busbars that, pre-compensation, exhibit the worst voltage deviation should be solely considered as candidates for the placement of compensation; these busbars are called *weak*. By reducing the number of possible locations for compensation the problem size will be reduced, and an improvement in convergence on the global optimum should be observed.

Another technique that could be exploited is cost analysis. In a multi-objective optimisation, where a cost is incurred (by installation of devices, for example) in an effort to reduce cost (from MVAr loss for example), it is possible that conclusions may be drawn in respect of the feasible reduction in cost that can be achieved. This information can then be used to limit the maximum amount that may be spent on the implementation of a solution. For the

purpose of explanation, consider the problem of SVC siting for power system loss reduction. If analysis of the uncompensated system reveals that it is only physically possible to incur a cost saving due to loss reduction of, say, 500 pounds, it would be unnecessary to consider a candidate solution that would cost more than 500 pounds to implement: no matter how beneficial it may be, it costs more than it saves.

Although problem space reduction is a powerful technique, it must be used with care and based on sound assertions, for fear of over confining the search and preventing the optimisation from finding global optima. For example, in the case of weak busbar analysis, it is conceivable that the optimal location for a device is *between* two weak busbars and not actually at either.

A technique for performing a GA search, utilising problem space reduction techniques, that will not accidentally exclude possible global optima, could be implemented by using this reduced problem space for seeding the population and, other than that, allow the search to continue unconfined. In this manner the search is initially steered to regions of potential optima, but the whole problem space is still available for exploration.

10.5 Solution performance caching

One improvement that GCP sorely needs is some form of solution caching. This technique is, on an immediate level, used purely to speed up the running time that GCP takes to complete optimisation. This is done by exploiting the fact that during optimisation, especially during the later stages, the same string may be evaluated many times. By storing or *caching* previous strings and their associated performance in a way that facilitates fast retrieval – hashing may well be a good direction for investigations – it may be possible to make a significant saving in time.

In the case of Beta study, investigated in this work, evaluation of a single string could take between a 0.5 and 1 seconds. When this is multiplied by the size of the population and the number of generations extremely long running times can result. Such problems, with long string-evaluations times, could well benefit from solution caching, although caution must be exercised as the caching operation itself can be time consuming.

Another interesting consequence of a "cached GA" is the large amount of data available at the end of the run. These data could well be useful, for example, for seeding a repeat run on the same or a similar problem.

10.6 Adaptive GA parameters

Finally, another long term goal could be the specification of ways of predicting the optimal value for the parameters of the GA, without the need for the lengthy investigations carried out in this work. This could well be an arduous task requiring thorough investigation of many different problems, but would significantly strengthen the algorithm.

10.7 Conclusion

This work has shown GAs to be a powerful tool for solving practical RCP problems, as demonstrated by the experiments conducted using GCP, the GA based RCP software developed in the course of this work. By implementing the concepts presented in this chapter, it is hoped that the performance and functionality of GCP can be further improved and enhanced, resulting in a more productive and useful package.

Bibliography

W.N.W. Abdullah, H Saibon, A.A. Mohd. Zain, and K.L. Lo. Optimal capacitor placement using genetic algorithm. In *IPEC1997*, pages 18–23, Singapore, May 1997.

S. Aboreshaid and R. Billinton. Probabilistic evaluation of voltage stability. *IEEE Transactions on Power Systems*, 14(1):342–348, February 1999.

S. Ahmed, G. Strbac, L.Z. Yao, A. Dixon, A. Chebbo, and A. Ekwue. Allocation of reactive support in a competitive environment. In *13th PSCC*, pages 368–374, Trondhiem, June 1999.

C. Aldridge, A. Chebbo, and H.B. Wan. Project progress meetings. Per. comm., 2001.

C.J Aldridge, K.P. Dahal, and J.R. McDonald. *Modern Optimisation Techniques in Power Systems*, volume 20 of *International series on microprocessor-based and intelligent systems engineering*, chapter Genetic Algorithms for Scheduling Generation and Maintenance in Power Systems. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

K. Aoki, W. Fan, and A. Nishikori. Optimal VAr planning by approximation method for recursive mixed-integer linear programming. *IEEE Transactions on Power Systems*, 3(4):1741–1747, November 1988.

F.H. Arnold, P.L. Wintrode, K. Miyazaki, and A. Gershenson. How enzymes adapt: Lessons from directed evolution. *Trends in Biochemical Sciences*, 26(2): 100–106, February 2001.

C.J. Bridenbaugh, D.A. DiMascio, and R D'Auila. Voltage control improvement through capacitor and transformer tap optimization. *IEEE Transactions on Power Systems*, 7(1):222–227, Feb 1992.

R.T. Byerly, D.T. Poznaniak, and E.R. Jr Taylor. Static reactive compensation for power transmission systems. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3997–4005, October 1982.

C.A. Cañizares and Z.T. Faur. Analysis of SVC and TCSC controllers in voltage collapse. *IEEE Transactions on Power Systems*, 14(1):158–165, February 1999.

C.S. Chang and J.S. Huang. Optimal multiobjective SVC planning for voltage stability enhancement. *IEE Proceedings - Generation, Transmission, and Distribution*, 145(2):203–209, March 1998.

F. de la C. Chard. *Electricity Supply: Transmission and Distribution.* Longman, 1976.

A.M. Chebbo, D.J. Coates, A.P. Malins, and K.M. Radi. Reactive power market and reactive plant control in England and Wales. In *CIGRE Symposium*, June 1999.

A.M. Chebbo and M.R. Irving. Combined active and reacive despatch. part 1: Problem formulation and solution algorithm. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(4):393–400, July 1995.

A.M. Chebbo, M.R. Irving, and N.H. Dandachi. Combined active and reacive despatch. part 2 : Test results. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(4):393–400, July 1995.

Y.L. Chen, C.W. Chang, and C.L. Liu. Efficient methods for identifying weak nodes in electrical power networks. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(3):317–322, May 1995.

Y.L. Chen and C.C. Liu. Multiobjective VAr planning using the goal attainment method. *IEE Proceedings - Generation, Transmission, and Distribution*, 141(3): 227–232, May 1994.

D.T.Y. Cheng and J.F. MacQueen. Some implimentation issues of a linear based reactive power optimisation program. In *12th Power Systems Computation Conference*, August 1996.

T.S. Chung and H.C. Leung. A genetic algorithm approach in optimal capacitor selection with harmonic distortion considerations. *Electrical Power and Energy Systems*, 21, 1999.

R.F. Cook. Analysis of capacitor application as affected by load cycle. *AIEE Transactions, pt. III-A Power Apparatus and Systems*, 72:950–957, October 1959.

R.F. Cook. Optimizing the application of shunt capacitors for reactive-volt-ampere control and loss reduction. *AIEE Transactions, pt. III-A, Power Apparatus and Systems*, pages 430–442, August 1961.

C. Darwin. *The Origin of Species*. Oxford Iniversity Press, 1859.

K. Deb and H.G. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):197–221, 2001.

M. Delfanti, G.P. Granelli, P. Marannino, and M. Montagna. Optimal capacitor placement using deterministic and genetic algorithms. *IEEE Transactions on Power Systems*, 15(3):1041–1046, August 2000.

C.L. DeMarco and T.J. Overbye. An energy based security measure for assessing vulnerability to voltage collapse. *IEEE Transactions on Power Systems*, 5(2):419–427, May 1990.

R. Dunn. CPF: Complex Power Flow. PSMAC, School of Electrical and Electronic Engineering, University of Bath, Claverton Down. Bath, 2001.

M.Z. EL-Sadek, M.M. Dessouky, G.A. Mahmoud, and W.I. Rashed. Combined use of tap-changing transformer and static var compensator for enhancement of steady-state voltage stabilities. *Electric Power Systems Research*, 45: 4755, 1998a.

M.Z. EL-Sadek, M.M. Dessouky, G.A. Mahmoud, and W.I. Rashed. Series capacitors combined with static var compensators for enhancement of steady-state voltage stabilities. *Electric Power Systems Research*, 44:137–143, 1998b.

I.A. Erinmez. The static synchronous compensator. MSc Course Special Intrest Programme, School of Electrical and Electronic Engineering, University of Bath, April 2000.

I.A. Erinmez and A.M. Foss. Static synchronous compensator (statcom). Technical report, Working Group 14.19, CIGRE Study Committee 14, September 1998.

L.L. Freris and A. M. Sasson. Investigation of the load-flow problem. *Proc. IEE*, 1968.

M.M. Gavrilovic, S Miske, and P.R Nannery. Bibliography of static VAr compensators. working group 79.2 on static VAr compensators in A.C. substations of the IEEE substations committee. *IEEE Transactions on Power Apparatus and Systems*, PAS-102(12):3744–3752, December 1983.

D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

L. Gyugyi. *Solid-State Synchronous Voltage Sources for Dynamic Compensation and Real-Time Control of AC Transmission Lines*. Emerging Practices In Technology. IEEE, March 1994.

E. Handschin, M. Heine, D. Knig, T. Nikodem, T. Seibt, and R. Palma. Object-oriented software engineering for transmission planning in open access. *IEEE Transactions on Power Systems*, 13(1):94–100, 1998.

H.H. Happ and K.A. Wirgau. Static and dynamic compensation in system planning. *IEEE Transactions on Power Apparatus and Systems*, PAS-97(5):1564–1577, Sep / Oct 1978.

R.L. Hauth, T Humann, and Newell. Application of a static VAr system to regulate system voltage in western nebraska. *PowAppSys*, PAS-97(5):1955–1964, Sep / Oct 1978.

R.L. Hauth, Jr. S.A. Miske, and F Nozari. The role and benefits of static VAr systems in high voltage power system applications. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3761–3770, October 1982.

J.M. Henderson, H.O. Simmons Jr, and J.B. Tice. Kilovar supply in bulk-power transmission systems. *AIEE Transactions, pt. III-A Power Apparatus and Systems*, 76:1344–1351, February 1958.

G.T. Heydt and W.M. Grady. A matrix method for optimal VAr siting. *IEEE Transactions on Power Apparatus and Systems*, PAS-94(4):1214–1220, July 1975.

I.A. Hiskens and D.J. Hill. Incorporation of SVCs into energy function methods. *PowSys*, 7(1):133–140, February 1992.

J.H. Holland. *Adaption in Natural and Artificial Systems.* The University of Mitchigan Press, Ann Arbor, 1975.

C. Horwill, M.H. Baker, and H.L. Thanawala. Static VAr compensator design and application for UK transmission system regulation. In *IEE International Conference on Advances in Power System Control, Operation and Management*, pages 129–134, November 1991.

K. Iba. Optimal VAr allocation by genetic algorithm. *IEEE Conference on Neural Networks*, pages 163–168, 1993.

W.S. Jwo, C.W. Liu, C.C. Liu, and Y.T. Hsiao. Hybrid expert system and simulated annealing approach to optimal reactive power planning. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(4):381–385, July 1995.

A. Kalyuzhny, G. Levitin, D. Elmakis, and H. Ben-Haim. System approach to shunt capacitor allocation in radial distribution systems. *Electric Power Systems Research*, 56:5160, 2000.

D Karlsson *et al.* Protection against voltage collapse. Technical Report 179, Electra, working group 34.08LiLi, 1998.

L.L. Lai and J.T. Ma. Improved genetic algorithms for optimal power flow under both normal and contingent operation states. *International Journal of Electrical Power & Energy Systems*, 19(5):287–292, 1997a.

L.L. Lai and J.T. Ma. New approach of using evolutionary programming to reactive power planning with network contingencies. *European Transactions on Electrical Power*, 7(3):211–216, May/June 1997b.

L.L. Lai and J.T. Ma. Application of evolutionary programming to reactive power planning - comparison with nonlinear programming approach. *IEEE Transactions on Power Systems*, 12(1), February 1998.

K.Y. Lee, X Bai, and Y.M. Park. Optimization method for reactive power planning using a modified simple genetic algorithm. *IEEE Transactions on Power Systems*, 10(4):1843–1850, November 1995.

K.Y. Lee, J.L. Ortiz, Y.M. Park, and L.G. Pond. An optimisation technique for reactive power planning of subtransmission network under normal operation. *IEEE Transactions on Power Systems*, PWRS-1(2):153–169, May 1986.

K.Y. Lee and F.F. Yang. Optimal reactive power planning using evolutionary algorithms: A comparative study for evolutionary programming, evolutionary strategy, genetic algorithm, and linear programming. *IEEE Transactions on Power Systems*, 13(1), Feb 1998.

G. Levitin, S. Mazal-Tov, and B. Reshef. Genetic algorithms for optimal planning of transmission system reactive power compensation under security constraints. In *9th Mediterranean Electrotechnical Conference*, pages 897–900, 1998.

B. Lewin. *Genes VI.* Oxford University Press, 1997.

F. Li. *Optimal economic operation of electrical power systems using genetic based algorithms.* PhD thesis, School of Electrical and Electronic Engineering,Liverpool John Moores University, 1996.

T.T. Lie and W. Deng. Optimal flexible AC transmission systems (facts) devices allocation. *Electrical Power & Energy Systems*, 19(2):125–134, 1997.

M. Luckett. Have VArs, can travel. *IEE Review*, pages 207–210, September 1999.

J.T. Ma and L.L. Lai. Application of genetic algorithm to optimal reactive power dispatch including voltage-dependent load models. In *Proc. of the 1995 IEEE International Conference on Evolutionary Computation*, pages 5–10, Perth, Australia, 1995.

J.T. Ma and L.L. Lai. A new genetic algorithm for optimal reactive power dispatch. *Engineering Inteligent Systems for Electrical Engineering and Communications*, 5(2):115–120, June 1997.

J.F. MacQueen, M.E. Bradley, and U. Bryan. Tools for optimising transmission system performance. In *Arc Workshop on Emerging Issues and Methods in the Restructuring of the Electric Power Industry*, Perth, Western Australia, Australia, July 1998.

K.R.C Mamandur and R.D. Chenoweth. Optimal control of reactive power flow for improvements in voltage profiles and for real power loss minimisation. *IEEE Transactions on Power Apparatus and Systems*, PAS-100(7):3185–3189, July 1981.

Y. Mansour, W. Xu, and C. Rinzin. SVC placement using critical modes of voltage instability. *IEEE Transactions on Power Systems*, 9(2):757–763, May 1994.

J.R.S. Mantovani, S.A.G. Modesto, and A.V. Garcia. VAr planning using genetic algorithm and linear programming. *IEE Proceedings - Generation, Transmission, and Distribution*, 148(3):257–262, May 2001.

H. Mori. Optimal allocation of FACTS devices in distribution systems. In *IEEE – Power and Energy Systems: Winter Meeting*, 2001.

K. Nara, Y. Hayashi, K. Ikeda, and T. Ashizawa. Application of tabu search to optimal placement of distributed generator. In *IEEE-PES 2001 Winter Meeting*, Columbus, Ohio, USA, January 2001.

National Grid. Seven year statement. http://www.nationalgrid.com/uk/library/documents/sys_01/index2.html (viewed on 30/05/01) or contact Transmission Development, The National Company Plc, National Grid House, Kirby Corner Road, Covetry, CV4 8JY, 2001. Last visited 3/10/2001.

Object Management Group. Unified modeling language (UML), version 1.3. http://www.omg.org/technology/documents/formal/uml.htm, 2001. Last visited 2/8/2001.

G Opoku. Optimal power system VAr planning. *IEEE Transactions on Power Systems*, 5(1):53–59, February 1990.

C.J. Parker, D Sutanto, and I.F. Morrison. Application of an optimization method for determining the reactive margin from voltage collapse in reac-

tive power planning. *IEEE Transactions on Power Systems*, 11(3):1473–1481, August 1996.

J.D. Pilgrim and F. Li. Genetic algorithms with dynamically variable string length for improved svc siting on intact and contingent power systems. In *UPEC' 2000*, Belfast, Northern Ireland, U.K., September 2000a. Queen's University Belfast.

J.D. Pilgrim and F. Li. Improved static VAr compensator siting on power systems using a ga with variable string length. In *GECCO-2000: Late-breaking papers*, Las Vegas, Nevada, USA, July 2000b.

J.D. Pilgrim, F. Li, and R.K. Aggarwal. SVCs and SVC siting techniques: Literature synopsis. In Lefley P.W., editor, *UPEC' 99*, pages 115–118, Department of Engineering, University of Leicester, Leicester, LE1 7RH, UK., 1999. University of Leicester.

J.D.. Rawn. *Biochemistry*. Neil Patterson Publishers, 1989.

C. Schauder, M. Gernhardt, E. Stacey, L. Gyugyi, T. W. Cease, A. Edris, and M. Wilhelm. TVA STATCON project: Design, installation, and commissioning. In *36th CIGR Session : 14-106*, 1996.

W.R. Thomas, A.M. Dixon, D.T.Y. Cheng, R.M. Dunnett, G. Schaff, and J.D. Thorp. Optimal reactive planning with security constraints. *IEEE Power Industry Computer Applications Conference*, pages 79–84, May 1995.

D. Thukaram and A. Lomi. Selection of static VAr compensator location and size for system voltage stability improvement. *Electric Power Systems Research*, 54:139150, 2000.

E. Vaahedi, J. Tamby, Y. Mansour, W. li, and D. Sun. Large scale voltage stability constrained optimal VAr planning and voltage stability applications using existing OPF/optimal VAr planning tools. *IEEE Transactions on Power Systems*, 14(1):65–74, Feb 1999.

W3.org. The extensible markup language. http://www.w3.org/XML/, 2001. Last visited 9/8/2001.

B.M. Weedy. *Electrical Power Systems*. John Wiley and Sons, 3rd revised edition, 1987.

Appendix A

Published Papers

The following pages include papers published or pending publication by the author and based on work contained in this thesis:

- **"SVCs and SVC Siting Techniques Literature Synopsis"** presented at the *34th Universities Power Engineering Conference* (UPEC), University of Leicester, Leicester, September 1999.
- **"Genetic Algorithms With Dynamically Variable String Length For Improved Svc Siting On Intact And Contingent Power Systems"** presented at the *35th Universities Power Engineering Conference* (UPEC), Queen's University, Belfast, September, 2000.
- **"Improved Static VAr Compensator Siting on Power Systems Using a GA with Variable String Length"** presented at the *Genetic and Evolutionary Computation Conference*, Las Vegas, Nevada, 2000

SVCS AND SVC SITING TECHNIQUES LITERATURE SYNOPSIS

J.D. Pilgrim[1], F Li[1], R.K.Aggarwal[1]

1. University of Bath, UK

ABSTRACT

This paper presents a synopsis of papers concerned with Static VAr Compensators (SVCs) and SVC siting techniques. The first section of this paper outlines the requirement for reactive compensation plant on transmission systems, and the types of plant available to satisfy this need. SVCs supply variable amounts of reactive power (lagging and leading VArs) with an extremely fast response time, thus making them ideal for performing a number of highly beneficial functions [1], including control of voltage variation, and power system stability improvement[2]. The specific roles of SVCs are examined and the various types available are outlined. The final topic covered in this paper is an examination of a selection of papers that highlight the various techniques currently being used in the field of optimal SVC placement on power systems.

REACTIVE POWER COMPENSATION

The quality and reliability of a power supply is ensured by maintaining the load bus voltages within the permissible limits. Any system configuration, or power demand changes, can result in higher or lower voltages in the system. This situation can be improved by either redispatch, or installation of reactive sources. [3]

Henderson et al [4] investigated the feasibility of supplying system VAr requirements from generating stations. They conclude by stating that the economics viability of transmitting VArs is severely degraded by increases in distance and KW loading, and that over even moderate distances, VAr support at a receiving-end bus is normally required.

To satisfy this requirement for locally generated VArs, a number of forms of compensation plant have been developed. Hauth et al [2] summarised the various forms of compensation available and the functions they are suited to.

- Shunt capacitors
  - Steady state voltage control.
- Shunt reactors
  - Steady state voltage control.
  - Reduction of switching surge overvoltages
- Series capacitors
  - Power transfer and stabilisation.
- Synchronous condensers
  - Steady state and dynamic voltage control.
  - Power transfer stabilisation.
- Static VAr Compensators
  - Steady state and dynamic voltage control.
  - Reactive power flow control.
  - Power transfer and stabilisation.

SVCs are versatile devices that can be used to improve the operation of a power system in several ways. Among the alternatives, SVCs are a common choice, due to their quick response and flexible reactive power compensation. Shunt capacitors/reactors are also commonly used.

The following section gives a detailed desciption of SVCs, their role in power systems, and the various types.

INTRODUCTION TO STATIC VAR COMPENSATORS

SVCs are integrated systems of static electrical components (e.g. capacitors, reactors, transformers, and switches) combined in such a manner to provide rapid continuously variable shunt reactive power compensation [1] [2], and as such, be used to improve the operation of a power system.

Description of a technique for modeling SVCs is provided by Hill(1992) [5], in addition to notes by the previous cited references. [1, 2, 4, 6, 7].

The Role Of SVCs

The main objectives of the installation SVCs on a bulk power system, are as follows [2].

- Reactive power flow control during the steady state to:
  - Minimise excessive system losses
  - Maintain desired voltage profile on transmission network
- Control voltage variation due to:
  - Daily load cycle
  - Repetitive impact loads such as arc furnaces (voltage flicker)
  - Synchronising power flow swings
  - Dynamic variations in HVDC converter P, Q
  - Load rejection
- Power system stability improvement to:
  - Maintain steady state power transfer capacity
  - Prevent transient instability
  - Prevent voltage instability or voltage collapse
  - Prevent oscillatory dynamic instability

Types Of SVC

There are a number of aspects for design classification of SVC plant. Output control can be by active feedback or entirely passive, i.e. an inherent characteristic of the static components (e.g., saturable reactors). In addition, switching can be achieved via thyristors, or conventional (i.e., non-solid state) switches. [2].

A typical SVC can be defined as having some form of active control, e.g., voltage regulator feed-back control,

and one or more thyristors switch assemblies to control output.When operating in the linear region of the steady state control characteristic, the thyristors control the current through the SVC reactors, and either thyristors or conventional switches to switch SVC capacitor banks.

There are a number of alternative SVC configurations. The different configurations can be categorised as follows.

- Thristor-controlled reactor (TCR), fixed capacitor (FC)
- Segmented TCR-FC
- 12-pulse TCR-FC
- High impedance thyristor controlled transformer (TCT)
- Thyristor switched capacitor (TSC), TCR
- Mechanically switched capacitors (MSC), TCR
- Saturable reactor (SR)

In analysis of the costs of the different configurations there are several contributing factors to the economics. In respect to the capital cost the most expensive components are the solid state devices, non-conventional reactor and non-conventional transformer designs, and HV and EHV harmonic filter banks. In addition to this Byerly suggests that operational and maintainance cost can be of greater importance in such analysis (the difference in losses between various SVC configuration are significant)[1].

More specific aspects of SVCs have been outlined by Canizares [6], who presents the details of the steady state models used in the analysis of the effect on SVCs on voltage collapse.

An extensive bibliography of pre 1982 compensation literature was assembled by Gavrilovic et al[7].

INTRODUCTION TO SVC PLACEMENT

Although SVCs can offer financial benefits to a power systems company, many feasible siting plans (in terms of number of units, type, size, location, etc.) are highly suboptimal. In addition to this, the problem space quickly becomes very complex as the size of the system increases; for a large system the problem space would be highly non-linear, multi-modal and discontinuous.

Much work has been carried out to develop optimization techniques that can operate in such an environment. The remainder of this paper looks at a selection of papers from the various categories of algorithm.

SVC SITING TECHNIQUES

The first paper describing a structured approach to the placement of reactive compensation was Cook in 1959 [8], who pointed out the undesirable effects of improperly placed compensation. The paper then presents an analysis of fixed capacitors on radial circuits with distributed loads. Particular emphasis is placed on an evaluation of the reduction in energy losses taking into account a periodic load cycle. In 1961, Cook extended the theory to include both fixed and/or switched capacitors [9].

Happ [10] presents a method for planning reactive compensation, notably including the " . . . newly developed variable shunt reactive control devices referred to here as Static VAr Control devices . . . ". The authors include a selection of papers covering various methods of reactive power allocation optimization. They divide these into nonlinear programming and linearised techniques (linear programming, integer programming, 0-1 programming, and dynamic programming). They also present a suitable model for a thyristor control static VAr device, which they use in conjunction with a linear programming technique.

In the same year Hauth et al. (1978) [11] published details of the "Application of a static VAr system to regulate system voltages in Western Nebraska", the site and rating of the VAr sources where chosen via a combination of repeated load flow studies and heuristic knowledge. Also, Bridenbaugh et al [12] examine the new VAr scheme for the Ohio electricity system. The papers describe the considerations of a utility system planner contemplating controllable shunt compensation for their systems.

In addition to the development of specific planning algorithms, there have been advances in generic aspects of the problem. Aoki et al [13] address the problem of the non- discrete aspect to the specification of VAr source installation, and propose an algorithm which treats capacitors as discretely adjustable devices in real scale systems. Chen et al [14] proposed a algorithm for simplifying the problem space of such problems by identifying weak-buses, which are far more likely to be candidates for reactive compensation.

Lee et al [15] divided the planning problem into a master problem (determination of yearly investment in reactive power compensation devices for expected load growth over a planning horizon), and two subproblems, P-Q (determine the optimal operation of the power system under normal conditions in respect to real and reactive power).

Opoku [16] uses a duplex simplex linear programming technique coupled with relaxation and contingency analysis. The method is applied to the IEEE 30-bus and 118-bus systems. Opoku outlines the modification necessary to promote convergence when dealing with large, highly stressed or contingent systems.

More recently, there have been many innovative techniques developed in the field of SVC expansion planning. Notably, most modern algorithms utilise some form of artificial intelligence.

Simulated annealing
Simulate annealing (SA) exploits the resemblance between a minimisation process and the cooling of molten metal; the energy of the molten is minimal when the cooling process is finished. The technique attempts to avoid entrapment in poor local minima by allowing occasional uphill movement [17][18].

Chang and Hung [19] formulate a scheme for optimal multiobjective SVC planning. Using parallel simulated annealing and a Lagrange multiplier the technique is concerned with optimising for voltage stability enhancement. They define a fuzzy performance index which represents system reactive power margin, system I^2R losses and voltage depressions at critical points. The siting is constrained in terms of the SVC characteristics, the number of SVCs and the overall capacity to be installed. The technique is tested on the IEEE 14-bus system, under normal and contingent conditions.

Chen and Liu [18] worked on a similar optimisation using a goal attainment method based on simulated annealing; optimising for active power loss reduction, minimisation of SVC investment cost, system security margin robustness and reduction of the voltage deviation of the system. They discuss in detail the formulation of many of the main objectives and constraints commonly involved in an SVC placement problem. The proposed technique is tested on the AEP 14-busbar system.

Jwo et al [17] use simulated annealing in conjunction with a hybrid expert system. Each objective function is replaced with a fuzzy goal by the decision maker, thus transforming the multi-objective optimization in a single min/max optimisation. The technique is tested using a modification of the IEEE 30-bus system. In comparison with SA, a slight cost reduction, and significant CPU time reduction, are demonstrated.

Evolutionary Algorithms

The growing trend over recent years has been the application of evolutionary algorithm based techniques to the siting problem. Iba [20] presents a siting technique using genetic algorithms (GAs). The search algorithm, based on the mechanics of natural selection and genetics. The GA uses an interbreeding/crossover algorithm that utilizes system topology and objective function heuristics, and also expert system stochastic if-then rules. The technique is applied to practical 51-bus and 224-bus systems.

Lai and Ma present two papers utilising an evolutionary based technique - Evolutionary Programming (EP). The technique optimises for total energy loss cost and VAr installation cost, and is applied to the IEEE 30-bus system. The first incorporates network contingencies [21]. The second presents a comparison with a nonlinear programming approach [22].

Lee and Yang [23] analyse the effectiveness of the three main evolutionary algorithms (EA's); evolutionary programming, evolutionary strategy, and genetic algorithms. They bench test these against a linear programming technique.

Finally Vaahedi et al [24], demonstrated that existing OPF/VAr planning tools can be accurately used to address voltage stability constrained VAr planning and voltage stability applications, in addition to the traditional feasibility criteria of acceptable voltage profile.

CONCLUSIONS

Due to the speed in response of SVCs, and the continuously variable nature of their reactive power output, SVC are the preferred reactive compensation plant on bulk transmission systems, in order to improve quality and reliability.

The financial benefits include more than reduced system losses; more efficient use of existing lines and plant will prolong their service life and possibly avoid the need for installment of new plant.

With changes in supply/demand patterns and increasing interconnection of systems, there is always a requirement for future reactive planning planning of power systems. The problem space for the planning activity quickly becomes very complex as the size of the system increases; for a large system the problem space would be highly non-linear, multi-modal and discontinuous. Finding optimal solutions for such a problem requires very sophisticated techniques. There are two growing trends in this field. Firstly, is the novel incorporation of artificial intelligence, secondly is in the development of combined, hybrid techniques. Methodologies drawing from evolutionary programming, simulated annealing, expert systems fuzzy logic, and neural-networks, can all contribute to solutions to this challenging optimisation problem.

ACKNOWLEDGEMENTS

The authors would like to thank the Engineering and Physical Sciences Research Council and The National Grid Company plc for their support in this research.

REFERENCES

[1] Byerly R.T., Poznaniak D.T., and Taylor E.R. Jr. Static Reactive Compensation for Power Transmission Systems. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3997-4005, October 1982.

[2] Hauth R.L., Miske Jr. S.A., and Nozari F. The Role and Benefits of Static VAR Systems in High Voltage Power System Applications. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3761-3770, October 1982.

[3] Weedy B.M. *Electrical Power Systems*. John Wiley and Sons, 3, rev. edition, 1987.

[4] Henderson J.M., Simmons Jr H.O., and Tice J.B. Kilovar Supply in Bulk-Power Transmission Systems. *AIEE Transactions, pt. III-A Power Apparatus and Systems*, 76:1344–1351, February 1958.

[5] Hill D.J. and Hiskens I.A. Incorporation of SVCs into Energy Function Methods. *PowSys*, 7(1):133-140, February 1992.

[6] Canizares C.A. and Faur Z.T. Analysis of SVC and TCSC controllers in Voltage Collapse. *IEEE*

Transactions on Power Systems, 14(1):158–165, February 1999.

[7] Gavrilovic M.M., Miske S, and Nannery P.R Chairman. Bibliography of Static VAR Compensators. Working Group 79.2 on Static VAR Compensators in A.C. Substations of the IEEE Substations Committee. *IEEE Transactions on Power Apparatus and Systems*, PAS 102(12):3744–3752, December 1983.

[8] Cook R.F. Analysis of Capacitor Application as Affected by Load Cycle. *AIEE Transactions, pt. III-A Power Apparatus and Systems*, 72:950–957, October 1959.

[9] Cook R.F. Optimizing the Application of Shunt Capacitors for Reactive-Volt-Ampere Control and Loss Reduction. *AIEE Transactions, pt. III-A, Power Apparatus and Systems*, pages 430–442, August 1961.

[10] Happ H.H. and Wirgau K.A. Static and Dynamic Compensation in System Planning. *IEEE Transactions on Power Apparatus and Systems*, PAS-97(5):1564–1577, Sep / Oct 1978.

[11] Hauth R.L., Humann T, and Newell. Application of a Static Var System to Regulate System Voltage in Western Nebraska. *PowAppSys*, PAS-97(5):1955–1964, Sep / Oct 1978.

[12] Bridenbaugh C.J., DiMascio D.A., and D'Auila R. Voltage Control Improvement Through Capacitor and Transformer Tap Optimization. *IEEE Transactions on Power Systems*, 7(1):222–227, Feb 1992.

[13] Aoki K., Fan W., and Nishikori A. Optimal VAr Planning by Approximation Method for Recursive Mixed-Integer Linear Programming. *IEEE Transactions on Power Systems*, 3(4):1741–1747, November 1988.

[14] Chen Y.L., Chang C.W., and Liu C.L. Efficient Methods for Identifying Weak Nodes in Electrical Power Networks. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(3):317–322, May 1995.

[15] Lee K.Y., Ortiz J.L., Park Y.M., and Pond L.G. An Optimisation Technique for Reactive Power Planning of Subtransmission Network Under Normal Operation. *IEEE Transactions on Power Systems*, PWRS-1(2):153–169, May 1986.

[16] Opoku G. Optimal Power System VAR Planning. *IEEE Transactions on Power Systems*, 5(1):53–59, February 1990.

[17] Jwo W.S., Liu C.W., Liu C.C., and Hsiao Y.T. Hybrid Expert System and Simulated Annealing Approach to Optimal Reactive Power Planning. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(4):381–385, July 1995.

[18] Chen Y.L. and Liu C.C. Multiobjective VAR Planning Using the Goal Attainment Method. *IEE Proceedings - Generation, Transmission, and Distribution*, 141(3):227–232, May 1994.

[19] Chang C.S. and Huang J.S. Optimal Multiobjective SVC Planning for Voltage Stability Enhancement. *IEE Proceedings - Generation, Transmission, and Distribution*, 145(2):203–209, March 1998.

[20] Iba K. Optimal VAr Allocation by Genetic Algorithm. *IEEE Conference on Neural Networks*, pages 163–168, 1993.

[21] Lai L.L. and Ma J.T. New Approach of Using Evolutionary Programming to Reactive Power Planning with Network Contingencies. *European Transactions on Electrical Power*, 7(3):211–216, May/June 1997.

[22] Lai L.L. and MA J.T. Application of Evolutionary Programming to Reactive Power Planning - Comparison with Nonlinear Programming Approach. *IEEE Transactions on Power Systems*, 12(1), February 1998.

[23] Lee K.Y. and Yang F.F. Optimal Reactive Power Planning Using Evolutionary Algorithms: A Comparative Study for Evolutionary Programming, Evolutionary Strategy, Genetic Algorithm, and Linear Programming. *IEEE Transactions on Power Systems*, 13(1), Feb 1998.

[24] Vaahedi E. et al. Large Scale Voltage Stability Constrained Optimal VAr Planning and Voltage Stability Applications Using Existing OPF/Optimal VAr Planning Tools. *PowSys*, 14(1):65–74, Feb 1999.

AUTHOR'S ADDRESS

The first author can be contacted at

Power and Energy Systems Group
Department of Electrical and Electronic Engineering
University of Bath
Claverton Down
Bath
UK
email: james@ee.bath.ac.uk

GENETIC ALGORITHMS WITH DYNAMICALLY VARIABLE STRING LENGTH FOR IMPROVED SVC SITING ON INTACT AND CONTINGENT POWER SYSTEMS

J.D. Pilgrim[1], F Li[1]
1. University of Bath, UK

ABSTRACT

A modified *Genetic Algorithm* (GA) with *Dynamically Variable String Length* (DVSL) is presented. The algorithm is applied to the problem of siting *Static VAr Compensators* (SVCs) on a power system, the objective of which is to produce a viable power system at the cheapest cost. The optimal number of SVCs to site, and their sizes, are not known in advance.
The ability of a simple GA to optimise this problem is reduced by using suboptimal length strings, which are either too short, and cannot represent the optimal siting plan, or too long, with high coding redundancy. The DVSL algorithm aims to remove this effect by optimising the string length during run time. This is achieved by examining the population at each specified generation, and, based on a set of if then rules, making adjustments to the population string length.
The algorithm is tested on the IEEE 30-bus power system under heavy (double) loads . The GA with DVSL is compared with a simple GA, demonstrating an improvement in the robustness of the optimisation in terms of its sensitivity to the initial string length, in finding the global optimum.

INTRODUCTION

For the operators of a bulk electrical power transmission system, maintaining quality and reliability is of the utmost importance. With changes in supply and demand patterns, and increasing interconnection of systems, there is a constant need for future planning of power systems. One aspect of this is ensuring sufficient reactive power compensation [1]. Amongst the alternatives, a common choice of compensation is the SVC.

Static VAr Compensator Siting

Reactive power is the imaginary component of complex power, and is closely related to the system voltages. Injecting inductive MVArs at a node acts to reduce the voltage at that point, whilst injecting capacitive MVArs at a node acts to increase the voltage. The quality and reliability of a bulk power transmission system can be ensured by maintaining the load bus voltages within their permissible limits; this can be achieved via the allocation of reactive power compensation [2].

The goal of reactive power planning is to determine the the most economical pattern of new reactive power sources (usually reactive compensation) which will be required for satisfactory operation of the transmission system in future years [1], that is, to ensure the reserves are going to be there if they are needed. The definition of satisfactory operation generally involves factors such as minimising power loss [3], minimising voltage deviation [4], ensuring load flow balance [5] and maximising voltage collapse margins [6] (maximising voltage stability). The resulting siting plan will be a set of new reactive sources each specified in terms of type, size and location.

The Static VAr Compensator (SVC) provides rapid and continuously variable amounts of shunt reactive power compensation within its operating range – which can cover inductive and capacitive MVAr requirements– and as such, can be used to improve the operation of a power system [7, 8]. Although SVCs can offer benefits to a power system operator, they are an expensive solution, and many feasible siting plans are highly suboptimal. In addition to this, the problem space quickly becomes very complex as the size of the system increases; for a large system the problem space would be highly non-linear, multi-modal and discontinuous [9].

Much work has been carried out to develop optimisation techniques that can operate in such an environment. Decomposition [10] and Linear Programming [11] have been used, as well as Simulated Annealing [12]. A growing trend over recent years has been the application of genetic algorithm based techniques to the siting problem. GAs have been shown to be powerful optimisation techniques, which, theoretically, converge to the global optimum with a probability of one [9, 13–15]. In these algorithms the maximum number of SVCs coded within a string needed to be specified before optimisation takes place. This paper shows how excessive string redundancy can effect the probability of the algorithm finding the optimum. The DVSL techniques aims to minimise this effect by optimising the string length during run time,

The Genetic Algorithm

Genetic Algorithms are search algorithms based on the mechanics of natural selection and natural genetics. The GA uses a *population* of binary strings; each string represents a point in the problem space: a possible solution to the problem.

At each time instant, or *generation*, the population is subjected to a number of *genetic operators*: *crossover*, *reproduction* and *mutation*. The genetic operators copy and exchange the information contained in the population to make new solutions. These new solutions can then be evaluated by the *fitness function* and the best solutions recorded.

GA perform a parallel search, using payoff information and probabilistic transition rules. These features make GAs robust, parallel algorithms to adaptively search for the global optimal point [16]. In addition, they are naturally integer based methods.

SVC SITING USING A GA

There are two important aspects to the application of a GA: evaluation of fitness and codification of strings.

Evaluation of Fitness
In the context of SVC siting, fitness evaluation involves the following tasks:

- Decoding the string into an SVC siting plan.
- Implementing the required modifications to the test systems: the intact network and the network with any contingencies.
- Performing load flow on the test systems.
- Calculating the associated fitness value using any required data, for example, data from the siting plan and results of load flow calculations.

This paper employs an objective function based on that used by the UK *National Grid Company* (NGC), in their reactive power planning software, SCORPION. SCORPION aims to minimise a cost function based on the total cost of reactive generation. This is formulated in terms of actual generators, which are assigned a low cost rate, and potential generators, which are assigned a high cost rate. The solution must also satisfy voltage and load flow constraints [1, 17].

The cost function used here can be expressed as follows:

$$C = I_Q * R_I + U_Q * R_U + U_{FQ} * R_{FQ} + U_{FP} * R_{FP} \quad (1)$$

where,

I_Q = installation of additional MVAr sources (MVAr)
U_Q = utilisation of existing MVAr sources (MVAr)
U_{FQ} = utilisation of fictitious MVArs (MVAr)
U_{FP} = utilisation of fictitious MWs (MW)

R_U = cost rate of existing MVAr sources (£/MVAr)
R_I = cost rate of additional MVAr sources (£/MVAr)
R_{FQ} = cost rate of fictitious MVArs (£/MVAr)
R_{FP} = cost rate of fictitious MWs (£/MW)

Utilisation of existing MVAr sources are assigned a low cost rate ($R_U = 1$), installing additional MVArs is assigned a high cost rate ($R_I = 100$), and the use of fictitious sources is assigned a very high cost rate ($R_{FQ} = 1e4$, $R_{FP} = 1e6$) which effectively acts as a load flow imbalance penalty function.

For the M members of the population, each cost value C_i is then mapped to fitness value F_i using,

$$F_i = 1 - \frac{C_i - C_{min}}{C_{max} - C_{min}} \quad (2)$$

with,

$$C_{max} = \{C_i | C_i \geq C_j \; \forall C_j, j = 1, \ldots, M\} \quad (3)$$
$$C_{min} = \{C_i | C_i \leq C_j \; \forall C_j, j = 1, \ldots, M\} \quad (4)$$

Codification of Strings
Choosing a suitable coding scheme is fundamental in implementing a robust GA. The simplest form of coding is *concurrent coding*: a string is the consolidation of a number of substrings, each representing a part of the solution.

The siting problem can be thought of as a pairing, or allocation, problem: there are a fixed number of possible host nodes and SVCs; a solution is a set of nodes and associated SVCs. Thus, each substring describes one node-SVC pair. There are two approaches to coding this type of problem: one approach is that the string contains a substring for each node of the system, describing any SVCs connected, and the other approach, for each SVC, describing to which node it is connected. The redundancy of a particular coding method is related to the relative sizes of the number of nodes and number of SVCs. If every node is likely to have an SVC connected, the first method would be an acceptable option. If, as is more often the case, the number of nodes is much greater than the optimal number of SVCs, the second method produces much shorter string lengths.

Using this second method, the population can be described as follows: The algorithm uses a population, P, of M strings.

$$P = \{s_1, \ldots, s_M\}$$

Each of these strings concurrently encodes L SVCs.

$s_i = \{c_1, \ldots, c_L\}$ $\quad s_i \in P$
L = number of SVCs
c_i = SVC substring

A single SVC substring describes the siting of one SVC, in terms of size, location and a status bit. An SVC is considered inactive (not to be installed), if it has a zero status bit, a size of zero, or an illegal location code.

This coding algorithm presents a problem: as the strings encode a fixed maximum number of SVCs (L), how can this upper limit be decided? One option is to have a sufficiently large string size so that the string can always encode the optimal number of SVCs – a substring for every candidate node – [18] but this, due to the added redundancy, can have a negative effect on the robustness of the GA, particularly when concidering realistic systems whcih could have upward of a thousand nodes.

The solution described here is to allow, between successive populations, the length of all strings in the population to change by the same amount.

GENETIC ALGORITHM WITH DYNAMICALLY VARIABLE STRING LENGTH

To overcome the problem of having to pre-specify an optimal string length for each problem, for example, each different power system, the GA with DVSL uses a high level operator to adjust the string length between successive generations.

The high-level operator

For each specified generation, the new high-level operator is called. Its purpose is to examine the population, and, using a rule set, decide whether the population string length should be adjusted. If required, it must perform the necessary operations to lengthen or shorten the strings.

String Length Adjustment Rules
The algorithm recieves a population, P, containing M strings. Each of these strings encode L SVCs.

For any one string, a certain number of SVCs will be active – that is, they will be installed on to the power system. This value will be called a_i, defined as the number of active SVCs described by the string s_i:

$$a_i = \text{number active SVCs in } s_i, \quad s_i \in P \tag{5}$$
$$a_i \leq L \tag{6}$$

Concidering the population as a whole, the values of a_{max} and a_{min} can then be defined:

$$a_{max} = \{a_i | a_i \geq a_j \forall a_j, j = 1, \ldots, M\} \tag{7}$$
$$a_{min} = \{a_i | a_i \leq a_j \forall a_j, j = 1, \ldots, M\} \tag{8}$$

By examining a_{max} and a_{min}, in relation to L, predictions can be made as to whether the string length is too short or too long. If $a_{max} << L$ or $a_{min} \approx L$ a smaller or longer string length, respectivly, would be suggested. These predictions have been mathamatical formulated in the form of the following if-then rules, which are used to determine when and how to adjust the string length:

$$if\,(a_{max} + L_{UM} < L)$$
$$then\ let\ (L' = a_{max} + L_{UM}) \tag{9}$$
$$if\,(a_{min} + L_{LM} > L')$$
$$then\ let\ (L'' = a_{min} + L_{LM}) \tag{10}$$

where,

L	= the string length in terms of the number of concurrently coded SVC substrings
a_{max}	= the maximum number of active SVCs in any one string of the population
a_{min}	= the minimum number of active SVCs in any one string of the population
L_{UM}	= the upper length margin
L_{LM}	= the lower length margin

The first rule is for *redundancy reduction*, and attempts to reduce the number of inactive SVCs by reducing the string length. The second rule is for *diversity preservation*, and attempts to allow the number of active SVCs to increase by increasing the string length.

The rules are evaluated sequentially, before any adjustments are made: the value of L' set in rule one, is used as the value of L' in rule two. Due to this, the second rule counteracts the effect of the first.

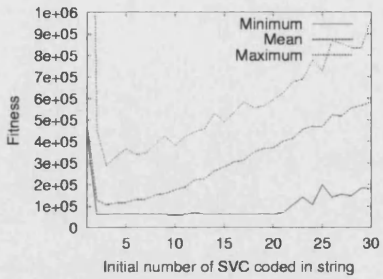

Figure 1: GA using static length strings

Once both rules have been evaluated, and the value of L'' has been quantified, each string in the population must be subjected to some form of transform, so that its length equals the required new string length. If $L'' < L$, the transformation is performed by truncating each string in the population down to the required length. This is effectivly making a new string by randomly resampling L'' substrings from the original string. If $L'' > L$, the transformation is performed by appending random binary digits to each string in the population up to the required length. This is effectively appending randomly intialised SVC substrings to the end of the string.

The values of L_{UM} and L_{LM} set the margin or tolerances of the two rules. Lower values of L_{UM}, the upper margin, increases the string shortening effect of rule 1, and higher values of L_{LM}, the lower margin, increase the string lengthening effect of rule 2.

RESULTS

All results presented concern the optimal placement of SVCs on the IEEE 30-bus test system: a standard test system for this type of problem, as described in [19]. Two cases are considered: case 1, the intact system under heavy loading, and case 2, a contingent system under heavy loading. The heavy load pattern and contingency are is the same as described in [9]: the loads are doubled , and lines 10-21 and 6-28 are contingent . In the following tests, the two cases, as described above, will be considered simultaneously, using a multiobjective function given by summing the cost (equation 1) for each system, but only concidering instalation cost once..

Simple Genetic Algorithm
Initially, a simple GA was implemented. Each nine bit SVC substring is interpreted as follows:

$\{b_0, \ldots, b_8\}$	= nine bit SVC substring
$\{b_0\}$	= status bit
$\{b_1, \ldots, b_5\}$	= location code
$\{b_6, \ldots, b_8\}$	= size code, zero to $\pm$ 700 MVArs

The GA optimisation parameters used were as follows:

- A population size of 20 strings was used.
- The strings were randomly initialised.
- There was a 0.01 chance of mutation.
- Ten reproduction operations were carried out for each generation.
- A single point crossover algorithm was used.
- The optimisation process stopped after no improvement was seen for 30 generations, up to a maximum of 100 generations.

Tests were carried out using strings that encoded various numbers of SVC substrings, L. For each string length the GA optimisation was performed 200 times, and the best fitness achieved in each run was recorded. Figure 1 shows a summary of the fitness values attained for each string length between $L = 1$ and $L = 30$, the maximum practical value for this 30-bus system. In figure 1, the string length is expressed in terms of the number of concurrently coded SVC substrings L. The figure shows the minimum, maximum and mean average values of those 200 fitness values. A fitness value of zero implies the resultant power system caused non-convergent load flow calculations.

In this problem, as described, the optimum siting plan is to install a 100MVArs SVC at node 28 and a 200MVArs SVC at node 9. This implies a string length of $L = 2$ is suitable for optimisation. As can be seen, the greatest chance of finding the optimal is for $L = 3$, implying a certain amount of redundancy is beneficial to the search process. As the string length increases, the chance of finding the global optimum decreases. After a point, the high redundancy of longer string lengths prohibited the GA from finding an optimal solution in any of the two hundred runs. In addition a string length of $L = 1$ prevented the string encoding the optimal solution, and only very expensive solutions were found. This high cost is actually associated with costs incurred due to load flow imbalance.

GA with DVSL

The GA was then modified so that it included the DVSL algorithm. An upper margin, L_{UM}, of zero, and a lower margin, L_{LM}, of three was used. The DVSL operator is called for every generation after the first. Figure 2 shows the results for a GA using DVSL. Results are shown, as before, for 200 runs of the GA at each initial string length, expressed in terms of the number of concurrently coded SVC substrings.

In these tests there appears to be no trend relating the initial string length to the chance of the GA finding the global optima. The DVSL algorithm prevents the GA suffering the negative effects of choosing a sub-optimal initial string length.

Comparison of the GA with and without DVSL

Figure 3 shows a comparison of the mean fitness values generated by the GA with and without DVSL, taken from figures 1 and 2. For initial string lengths between two and four SVC substrings, both algorithms perform equally reliably – although the simple GA never

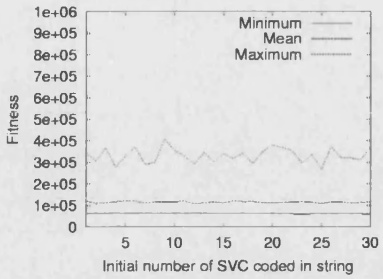

Figure 2: GA using Variable string length

outperforms the DVSL GA. The the two algorithms depart at $L = 5$ as the simple GA steadily increase the mean cost of the best solutions it can find.

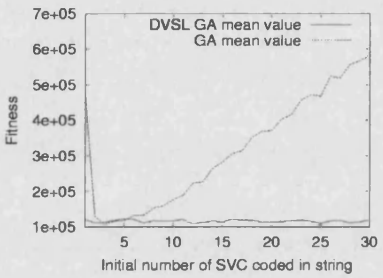

Figure 3: Comparison between GA with variable and static string length

CONCLUSION

This paper presents an algorithm for improved SVC siting using a Genetic Algorithm with Dynamically Variable String Length. The DVSL algorithm uses a rule set to adjust the GA string length between successive populations.

The technique is demonstrated for multiple SVC siting under heavily loaded intact and contingent conditions on the IEEE 30-bus test system.

The results presented demonstrate how the use of a Dynamically Variable String Length GA can significantly improve the robustness of the GA in terms of its sensitivity to the initial string length.

ACKNOWLEDGEMENTS

This research is funded by the Engineering and Physical

Sciences Research Council and The National Grid Company plc. Many thanks to Ahmad Chebbo and Chris Aldridge, of NGC, for their invaluable and continuing support for this project.

REFERENCES

[1] W.R. Thomas, A.M. Dixon, D.T.Y. Cheng, R.M. Dunnett, G. Schaff, and J.D. Thorp. Optimal Reactive Planning with Security Constraints. *IEEE Power Industry Computer Applications Conference*, pages 79–84, May 1995.

[2] B.M. Weedy. *Electrical Power Systems*. John Wiley and Sons, 3rd revised edition, 1987.

[3] W. Hubbi and T. Hiyama. Placement of Static VAR Compensators to Minimize Power System Losses. *Electric Power Systems Research*, 47(2):95–99, 1998.

[4] W.S. Jwo, C.W. Liu, C.C. Liu, and Y.T. Hsiao. Hybrid Expert System and Simulated Annealing Approach to Optimal Reactive Power Planning. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(4):381–385, July 1995.

[5] K.Y. Lee, X Bai, and Y.M. Park. Optimization Method for Reactive Power Planning Using a Modified Simple Genetic Algorithm. *IEEE Transactions on Power Systems*, 10(4):1843–1850, November 1995.

[6] Y. Mansour, W. Xu, and C. Rinzin. SVC Placement Using Critical Modes of Voltage Instability. *IEEE Transactions on Power Systems*, 9(2):757–763, May 1994.

[7] R.T. Byerly, D.T. Poznaniak, and E.R. Jr Taylor. Static Reactive Compensation for Power Transmission Systems. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3997–4005, October 1982.

[8] R.L. Hauth, Jr. S.A. Miske, and F Nozari. The Role and Benefits of Static VAR Systems in High Voltage Power System Applications. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3761–3770, October 1982.

[9] L.L. Lai and J.T. Ma. New Approach of Using Evolutionary Programming to Reactive Power Planning with Network Contingencies. *European Transactions on Electrical Power*, 7(3):211–216, May/June 1997.

[10] K.Y. Lee, J.L. Ortiz, Y.M. Park, and L.G. Pond. An Optimisation Technique for Reactive Power Planning of Subtransmission Network Under Normal Operation. *IEEE Transactions on Power Systems*, PWRS-1(2):153–169, May 1986.

[11] G Opoku. Optimal Power System VAR Planning. *IEEE Transactions on Power Systems*, 5(1):53–59, February 1990.

[12] C.S. Chang and J.S. Huang. Optimal Multiobjective SVC Planning for Voltage Stability Enhancement. *IEE Proceedings - Generation, Transmission, and Distribution*, 145(2):203–209, March 1998.

[13] K. Iba. Optimal VAr Allocation by Genetic Algorithm. *IEEE Conference on Neural Networks*, pages 163–168, 1993.

[14] K.Y. Lee and F.F. Yang. Optimal Reactive Power Planning Using Evolutionary Algorithms: A Comparative Study for Evolutionary Programming, Evolutionary Strategy, Genetic Algorithm, and Linear Programming. *IEEE Transactions on Power Systems*, 13(1), Feb 1998.

[15] A.J. Urdaneta, J.F. Gomez, E. Sorrentino, L. Flores, and R. Diaz. A Hybird Genetic Algorithm for Optimal Rective Power Planning Based Upon Successive Linear Programming. *IEEE Transactions on Power Systems*, 14(4):1292–1298, 1999.

[16] J.T. Ma and L.L. Lai. A New Genetic Algorithm for Optimal Reactive Power Dispatch. *Engineering Inteligent Systems for Electrical Engineering and Communications*, 5(2):115–120, June 1997.

[17] D.T.Y. Cheng and J.F. Macqueen. Some Implimentation Issues of a Linear Based Reactive Power Optimisation Program. In *12th Power Systems Computation Conference*, August 1996.

[18] G. Levitin, S. Mazal-Tov, and B. Reshef. Genetic Algorithms for Optimal Planning of Transmission System Reactive Power Compensation Under Security Constraints. In *9th Mediterranean Electrotechnical Conference*, pages 897–900, 1998.

[19] K.Y. Lee, Y.M. Park, and J.L. Ortiz. A United Approach to Optimal Real and Reactive Power Dispatch. *IEEE Transactions on Power Apparatus and Systems*, PAS-104:1147–1153, 1985.

AUTHOR'S ADDRESS

The first author can be contacted at

Power and Energy Systems Group
Department of Electrical and Electronic Engineering
University of Bath
Claverton Down
Bath
UK
email: james@ee.bath.ac.uk

(RWA) Improved Static VAr Compensator Siting on Power Systems Using a Genetic Algorithm with Dynamically Variable String Length

James D. Pilgrim
University of Bath
Claverton Down
BATH, U.K., BA2 7AY
james@ee.bath.ac.uk
(+44) 1225 826753

Furong Li
University of Bath
Claverton Down
BATH, U.K., BA2 7AY
f.li@bath.ac.uk
(+44) 1225 826416

Abstract

A modified *Genetic Algorithm* (GA) with *Dynamically Variable String Length* (DVSL) is presented. The algorithm is applied to the problem of siting *Static VAr Compensators* (SVCs) on a power system, the objective of which is to produce a viable power system at the cheapest cost. The optimal number of SVCs to site, and their sizes, are not known in advance.

The ability of a simple GA to optimise this problem is reduced by using suboptimal length strings, which are either too short, and cannot represent the optimal siting plan, or too long, with high coding redundancy. The DVSL algorithm aims to remove this effect by optimising the string length during run time. This is achieved by examining the population at each specified generation, and, based on a set of if-then rules, making adjustments to the population string length.

The algorithm is tested on the IEEE 30-bus power system under heavy (double) loads, for normal (intact) and contingent conditions. The GA with DVSL is compared with a simple GA, demonstrating an improvement in the robustness of the optimisation in terms of its sensitivity to the initial string length, in finding the global optimum.

1 Introduction

For the operators of a bulk electrical power transmission system, maintaining quality and reliability is of the utmost importance. With changes in supply and demand patterns, and increasing interconnection of systems, there is a constant need for future planning of power systems. One aspect of this is ensuring sufficient reactive power compensation (Thomas et al., 1995). Amongst the alternatives, a common choice of compensation is the SVC.

1.1 Static VAr Compensator Siting

Reactive power is the imaginary component of complex power, and is closely related to the system voltages. Injecting inductive MVArs at a node acts to reduce the voltage at that point, whilst injecting capacitive MVArs at a node acts to increase the voltage. The quality and reliability of a bulk power transmission system can be ensured by maintaining the load bus voltages within their permissible limits; this can be achieved via the allocation of reactive power compensation (Weedy, 1987).

The goal of reactive power planning is to determine the the most economical pattern of new reactive power sources (usually reactive compensation) which will be required for satisfactory operation of the transmission system in future years (Thomas et al., 1995), that is, to ensure the reserves are going to be there if they are needed. The definition of satisfactory operation generally involves factors such as minimising power loss Hubbi and Hiyama (1998), minimising voltage deviation (Jwo et al., 1995), ensuring load flow balance (Lee et al., 1995) and maximising voltage collapse margins (Mansour et al., 1994) (maximising voltage stability). The resulting siting plan will be a set of new reactive sources each specified in terms of type, size and location.

The Static VAr Compensator (SVC) provides rapid and continuously variable amounts of shunt reactive power compensation within its operating range – which can cover inductive and capacitive MVAr requirements– and as such, can be used to improve the operation of a power system (Byerly et al., 1982; Hauth et al., 1982). Although SVCs can offer benefits to a power system operator, they are an expensive solution, and many feasible siting plans are highly suboptimal. In addition to this, the problem space quickly becomes very complex as the size of the system increases; for a large system the problem space would be

highly non-linear, multi-modal and discontinuous (Lai and Ma, 1997).

Much work has been carried out to develop optimisation techniques that can operate in such an environment. Decomposition (Lee et al., 1986) and Linear Programming (Opoku, 1990) have been used, as well as Simulated Annealing (Chen and Liu, 1994; Chang and Huang, 1998). A growing trend over recent years has been the application of genetic algorithm based techniques to the siting problem. GAs have been shown to be powerful optimisation techniques, which, theoretically, converge to the global optimum with a probability of one (Lai and Ma, 1997; Iba, 1993; Lee and Yang, 1998; Miu et al., 1997; Urdaneta et al., 1999). In these algorithms the maximum number of SVCs coded within a string needed to be specified before optimisation takes place. This paper shows how excessive string redundancy can effect the probability of the algorithm finding the optimum. The DVSL techniques aims to minimise this effect by optimising the string length during run time,

1.2 The Genetic Algorithm

Genetic Algorithms are search algorithms based on the mechanics of natural selection and natural genetics.

The GA uses a *population* of binary strings; each string represents a point in the problem space: a possible solution to the problem.

Each string is evaluated by some form of *fitness function* which assigns each string a value that represents its fitness to solve the problem. As a GA is inherently a maximisation algorithm, higher fitness values indicate better solutions.

At each time instant, or *generation*, the population is subjected to a number of *genetic operators*: *crossover*, *reproduction* and *mutation*.

The reproduction operator represents a *survival of the fittest* mechanism. From the population at time t, a new population is selected. To select the $t + 1$ population, a fitness-weighted random algorithm is used: the fitter a particular string, the higher the probability it will be selected.

Crossover models the exchange of genetic information that occurs within a population. The strings of a population are randomly paired off; randomly selected adjacent sections of the strings of each pair are then removed and exchanged between the two.

The natural effect of errors occurring in genetic code – mutation – is modelled as follows: any bit copying operation is subjected to a probabilistic chance of a copying error. The chance of mutation occurring is small, but has been shown to be essential in the operation of the GA (Goldberg, 1989).

Once fitness values have been calculated for the new population, and any solutions from the current population have been recorded – best-so-far and so on – the next generation can be calculated by repeating the genetic operators.

Iteration is continued until some stop criteria is reached, which is usually after reaching a certain number of iterations or failing to improve for a number of iterations.

The features that set GAs apart from other optimisation techniques are,

- GAs perform a parallel search: the GA uses a population of independent points, not a single point. This population can move over hills and valleys, allowing global optimisation.
- GAs use payoff information (fitness or objective functions) directly for the search direction, not derivatives or other auxiliary knowledge. GAs can therefore deal with non-smooth, discontinuous and non-differentiable functions. This property relives GAs of approximation assumptions which are often required for other optimisation methods.
- GAs use probabilistic transition rules, not deterministic rules, to select the next generation. This enables them to search complicated and uncertain areas to find the global optimum, thus making them more flexible and robust than conventional methods.

These features make GAs robust, parallel algorithms to adaptively search for the global optimal point (Ma and Lai, 1997). In addition, they are naturally integer based methods.

2 SVC Siting Using a Genetic Algorithm

There are two important aspects to the application of a GA: evaluation of fitness and codification of strings.

2.1 Evaluation of Fitness

In the context of SVC siting, fitness evaluation involves the following tasks:

- Decoding the string into an SVC siting plan.
- Implementing the required modifications to the test systems: the intact network and the network with any contingencies.
- Performing load flow on the test systems.
- Calculating the associated fitness value using any required data, for example, data from the siting plan and results of load flow calculations.

This paper employs an objective function based on that used by the UK *National Grid Company* (NGC), in their reactive power planning software, SCORPION. SCORPION aims to minimise a cost function based on the total cost of reactive generation. This is formulated in terms of actual generators, which are assigned a low cost rate, and potential generators, which are assigned a high cost rate. The solution must also satisfy voltage and load flow constraints (Thomas et al., 1995; Cheng and Macqueen, 1996).

The cost function used here can be expressed as follows:

$$C = I_Q * R_I + U_Q * R_U + U_{FQ} * R_{FQ} + U_{FP} * R_{FP} \quad (1)$$

where,

U_Q = utilisation of existing MVAr sources (MVAr)
I_Q = installation of additional MVAr sources (MVAr)
U_{FQ} = utilisation of fictitious MVArs (MVAr)
U_{FP} = utilisation of fictitious MWs (MW)

R_U = cost rate of existing MVAr sources (£/MVAr)
R_I = cost rate of additional MVAr sources (£/MVAr)
R_{FQ} = cost rate of fictitious MVArs (£/MVAr)
R_{FP} = cost rate of fictitious MWs (£/MW)

Utilisation of existing MVAr sources are assigned a low cost rate ($R_U = 1$), installing additional MVArs is assigned a high cost rate ($R_I = 100$), and the use of fictitious sources is assigned a very high cost rate ($R_{FQ} =$ 1e4, $R_{FP} =$ 1e6) which effectively acts as a load flow imbalance penalty function.

To simultaneously optimise for a number of system states – for example, the intact system and a number of contingent states – the cost function needs to be reformulated into a multiobjective function. The total cost incurred by a particular siting strategy will be the sum of the cost incurred for each case considered, but any costs attributed to installation costs will only be considered once. Explicitly, optimising for N_{cases} cases, the cost function will be,

$$C = I_Q * R_I + \sum_{i=0}^{N_{cases}} (U_{Qi} * R_U + U_{FQi} * R_{FQ} + U_{FPi} * R_{FP}) \quad (2)$$

where U_{Qi}, U_{FQi} and U_{FPi} are the values of U_Q, U_{FQ} and U_{FP}, respectively, from case i.

For the M members of the population, each cost value C_i is then mapped to fitness value F_i using,

$$F_i = 1 - \frac{C_i - C_{min}}{C_{max} - C_{min}} \quad (3)$$

with,

$$C_{max} = \{C_i | C_i \geq C_j \ \forall C_j, j = 1, \ldots, M\} \quad (4)$$
$$C_{min} = \{C_i | C_i \leq C_j \ \forall C_j, j = 1, \ldots, M\} \quad (5)$$

2.2 Codification of Strings

Choosing a suitable coding scheme is fundamental in implementing a robust GA. The simplest form of coding is *concurrent coding*: a string is the consolidation of a number of substrings, each representing a part of the solution.

The siting problem can be thought of as a pairing, or allocation, problem: there are a fixed number of possible host nodes and SVCs; a solution is a set of nodes and associated SVCs. Thus, each substring describes one node-SVC pair. There are two approaches to coding this type of problem: one approach is that the string contains a substring for each node of the system, describing any SVCs connected, and the other approach, for each SVC, describing to which node it is connected. The redundancy of a particular coding method is related to the relative sizes of the number of nodes and number of SVCs. If every node is likely to have an SVC connected, the first method would be an acceptable option. If, as is more often the case, the number of nodes is much greater than the optimal number of SVCs, the second method produces much shorter string lengths.

Using this second method, the population can be described as follows: The algorithm uses a population, P, of M strings.

$$P = \{s_1, \ldots, s_M\}$$

Each of these strings concurrently encodes L SVCs.

$s_i = \{c_1, \ldots, c_L\}$ $\quad s_i \in P$
L = number of SVCs
c_i = SVC substring

A single SVC substring describes the siting of one SVC, in terms of size, location and a status bit. An SVC is considered inactive (not to be installed), if it has a zero status bit, a size of zero, or an illegal location code.

This coding algorithm presents a problem: as the strings encode a fixed maximum number of SVCs (L), how can this upper limit be decided? One option is to have a sufficiently large string size so that the string can always encode the optimal number of SVCs – a substring for every candidate node – Levitin et al. (1998) but this, due to the added redundancy, can have a negative effect on the robustness of the GA, particularly when concidering realistic systems whcih could have upward of a thousand nodes.

The solution described here is to allow, between successive populations, the length of all strings in the population to change by the same amount.

3 Genetic Algorithm with Dynamically Variable String Length

To overcome the problem of having to pre-specify an optimal string length for each problem, for example, each different power system, the GA with DVSL uses a high level operator to adjust the string length between successive generations.

3.1 The high-level operator

For each specified generation, the new high level operator is called. Its purpose is to examine the population, and, using a rule set, decide whether the population string length should be adjusted. If required, it must perform the necessary operations to lengthen or shorten the strings.

3.2 String Length Adjustment Rules

The algorithm recieves a population, P, containing M strings. Each of these strings encode L SVCs.

For any one string, a certain number of SVCs will be active – that is, they will be installed on to the power system. This value will be called a_i, defined as the number of active SVCs described by the string s_i:

$$a_i = \text{number active SVCs in } s_i, \quad s_i \in P \tag{6}$$

$$a_i \leq L \tag{7}$$

Concidering the population as a whole, the values of a_{max} and a_{min} can then be defined:

$$a_{max} = \{a_i | a_i \geq a_j \forall a_j, j = 1, \ldots, M\} \tag{8}$$

$$a_{min} = \{a_i | a_i \leq a_j \forall a_j, j = 1, \ldots, M\} \tag{9}$$

By examining a_{max} and a_{min}, in relation to L, predictions can be made as to whether the string length is too short or too long. If $a_{max} << L$ or $a_{min} \approx L$ a smaller or longer string length, respectivly, would be suggested. These predictions have been mathamatical formulated in the form of the following if-then rules, which are used to determine when and how to adjust the string length:

$$if\,(a_{max} + L_{UM} < L) \quad then\ let\ (L' = a_{max} + L_{UM}) \tag{10}$$

$$if\,(a_{min} + L_{LM} > L') \quad then\ let\ (L'' = a_{min} + L_{LM}) \tag{11}$$

where,

L	= the string length in terms of the number of concurrently coded SVC substrings
a_{max}	= the maximum number of active SVCs in any one string of the population
a_{min}	= the minimum number of active SVCs in any one string of the population
L_{UM}	= the upper length margin
L_{LM}	= the lower length margin

The first rule is for *redundancy reduction*, and attempts to reduce the number of inactive SVCs by reducing the string length. The second rule is for *diversity preservation*, and attempts to allow the number of active SVCs to increase by increasing the string length.

The rules are evaluated sequentially, before any adjustments are made: the value of L' set in rule one, is used as the value of L' in rule two. Due to this, the second rule counteracts the effect of the first.

Once both rules have been evaluated, and the value of L'' has been quantified, each string in the population must be subjected to some form of transform, so that its length equals the required new string length. If $L'' < L$, the transformation is performed by truncating each string in the population down to the required length. This is effectivly making a new string by randomly resampling L'' substrings from the original string. If $L'' > L$, the transformation is performed by appending random binary digits to each string in the population up to the required length. This is effectively appending randomly intialised SVC substrings to the end of the string.

The values of L_{UM} and L_{LM} set the margin or tolerances of the two rules. Lower values of L_{UM}, the upper margin, increases the string shortening effect of rule 1, and higher values of L_{LM}, the lower margin, increase the string lengthening effect of rule 2.

4 Results

All results presented concern the optimal placement of SVCs on the IEEE 30-bus test system: a standard test system for this type of problem, as described in (Lee et al., 1985). Two cases are considered: case 1, the intact system under heavy loading, and case 2, a contingent system under heavy loading. The heavy load pattern and contingency are the same as described in (Lai and Ma, 1997): the loads are doubled, and lines 10-21 and 6-28 are contingent. In the following tests, the two cases, as described above, will be considered simultaneously, using the multiobjective cost function given in equation 2.

4.1 Simple Genetic Algorithm

Initially, a simple GA was implemented. Each nine bit SVC substring is interpreted as follows:

$\{b_0, \ldots, b_8\}$	= nine bit SVC substring
$\{b_0\}$	= status bit
$\{b_1, \ldots, b_5\}$	= location code
$\{b_6, \ldots, b_8\}$	= size code, zero to $\pm$ 700 MVArs

The GA optimisation parameters used were as follows:

- A population size of 20 strings was used.
- The strings were randomly initialised.
- There was a 0.01 chance of mutation.
- Ten reproduction operations were carried out for each generation.
- A single point crossover algorithm was used.
- The optimisation process stopped after no improvement was seen for 30 generations, up to a maximum of 100 generations.

Tests were carried out using strings that encoded various numbers of SVC substrings, L. For each string length the GA optimisation was performed 200 times, and the best fitness achieved in each run was recorded. Figure 1 shows a summary of the fitness values attained for each string length between $L = 1$ and $L = 30$, the maximum practical value for this 30-bus system. In figure 1, the string length is expressed in terms of the number of concurrently coded SVC substrings L. The figure shows the minimum, maximum and mean average values of those 200 fitness values. A fitness value of zero implies the resultant power system caused non-convergent load flow calculations.

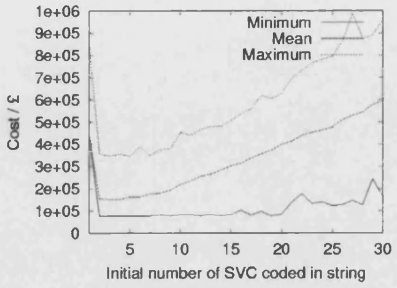

Figure 1: GA using static length strings

In this problem, as described, the optimum siting plan is to install a 100MVArs SVC at node 28 and a 200MVArs SVC at node 9. This implies a string length of $L = 2$ is suitable for optimisation. As can be seen, the greatest chance of finding the optimal is for $L = 3$, implying a certain amount of redundancy is beneficial to the search process. As the string length increases, the chance of finding the global optimum decreases. After a point, the high redundancy of longer string lengths prohibited the GA from finding an optimal solution in any of the two hundred runs. In addition a string length of $L = 1$ prevented the string encoding the optimal solution, and only very expensive solutions were found. This high cost is actually associated with costs incurred due to load flow imbalance.

4.2 GA with DVSL

The GA was then modified so that it included the DVSL algorithm. An upper margin, L_{UM}, of zero, and a lower margin, L_{LM}, of three was used. The DVSL operator is called for every generation after the first. Figure 2 shows the results for a GA using DVSL. Results are shown, as before, for 200 runs of the GA at each initial string length, expressed in terms of the number of concurrently coded SVC substrings.

In these tests there appears to be no trend relating the initial string length to the chance of the GA finding the global optima. The DVSL algorithm prevents the GA suffering the negative effects of choosing a sub-optimal initial string length.

4.3 Comparison of the GA with and without DVSL

Figure 3 shows a comparison of the mean fitness values generated by the GA with and without DVSL, taken from

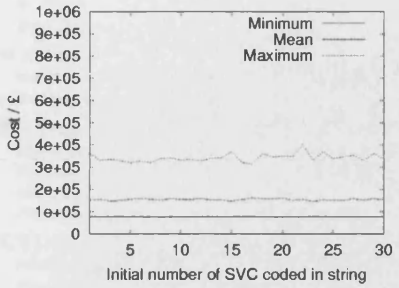

Figure 2: GA using Variable string length

figures 1 and 2. For initial string lengths between two and four SVC substrings, both algorithms perform equally reliably – although the simple GA never outperforms the DVSL GA. The the two algorithms depart at $L = 5$ as the simple GA steadily increase the mean cost of the best solutions it can find.

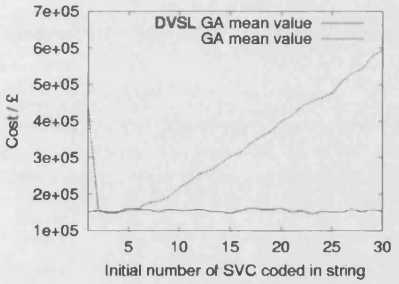

Figure 3: Comparison between GA with variable and static string length

4.4 Example of a Typical Run of the GA with DVSL

Figure 4 is a graphical display of a typical optimisation in one run of the GA with DVSL. The best fitness achieved in each generation and current string length are shown as functions of time, expressed in terms of the population generation. A long initial string length was used: 30 concurrently coded SVC substrings. Initially the string length converges on a much reduced string length. As the population converges, adjustments to the string length tends to be much smaller. Once the string length settles, the global optimum is found. The GA continues normally until no improvement is seen for the specified 30 generation before stopping.

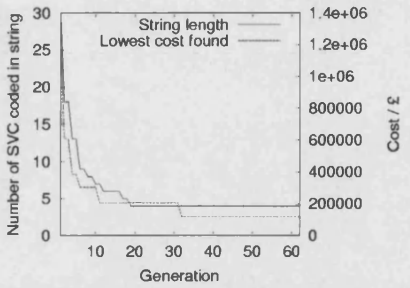

Figure 4: A typical run of the GA with DVSL

5 Conclusion

This paper presents an algorithm for improved SVC siting using a Genetic Algorithm with Dynamically Variable String Length. The DVSL algorithm uses a rule set to adjust the GA string length between successive populations.

The technique is demonstrated for multiple SVC siting under heavily loaded intact and contingent conditions on the IEEE 30-bus test system.

The results presented demonstrate how the use of a Dynamically Variable String Length GA can significantly improve the robustness of the GA in terms of its sensitivity to the initial string length.

6 Acknowledgements

This research is funded by the Engineering and Physical Sciences Research Council and The National Grid Company plc. Many thanks to Ahmad Chebbo and Chris Aldridge, of NGC, for their invaluable and continuing support for this project.

References

W.R. Thomas, A.M. Dixon, D.T.Y. Cheng, R.M. Dunnett, G. Schaff, and J.D. Thorp. Optimal reactive planning with security constraints. *IEEE Power Industry Computer Applications Conference*, pages 79–84, May 1995.

B.M. Weedy. *Electrical Power Systems*. John Wiley and Sons, 3rd revised edition, 1987.

W. Hubbi and T. Hiyama. Placement of static var compensators to minimize power system losses. *Electric Power Systems Research*, 47(2):95–99, 1998.

W.S. Jwo, C.W. Liu, C.C. Liu, and Y.T. Hsiao. Hybrid expert system and simulated annealing approach to optimal reactive power planning. *IEE Proceedings - Generation, Transmission, and Distribution*, 142(4):381–385, July 1995.

K.Y. Lee, X Bai, and Y.M. Park. Optimization method for reactive power planning using a modified simple genetic algorithm. *IEEE Transactions on Power Systems*, 10(4): 1843–1850, November 1995.

Y. Mansour, W. Xu, and C. Rinzin. Svc placement using critical modes of voltage instability. *IEEE Transactions on Power Systems*, 9(2):757–763, May 1994.

R.T. Byerly, D.T. Poznaniak, and E.R. Jr Taylor. Static reactive compensation for power transmission systems. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3997–4005, October 1982.

R.L. Hauth, Jr. S.A. Miske, and F Nozari. The role and benefits of static var systems in high voltage power system applications. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(10):3761–3770, October 1982.

L.L. Lai and J.T. Ma. New approach of using evolutionary programming to reactive power planning with network contingencies. *European Transactions on Electrical Power*, 7(3):211–216, May/June 1997.

K.Y. Lee, J.L. Ortiz, Y.M. Park, and L.G. Pond. An optimisation technique for reactive power planning of subtransmission network under normal operation. *IEEE Transactions on Power Systems*, PWRS-1(2):153–169, May 1986.

G Opoku. Optimal power system var planning. *IEEE Transactions on Power Systems*, 5(1):53–59, February 1990.

Y.L. Chen and C.C. Liu. Multiobjective var planning using the goal attainment method. *IEE Proceedings - Generation, Transmission, and Distribution*, 141(3):227–232, May 1994.

C.S. Chang and J.S. Huang. Optimal multiobjective svc planning for voltage stability enhancement. *IEE Proceedings - Generation, Transmission, and Distribution*, 145(2):203–209, March 1998.

K. Iba. Optimal var allocation by genetic algorithm. *IEEE Conference on Neural Networks*, pages 163–168, 1993.

K.Y. Lee and F.F. Yang. Optimal reactive power planning using evolutionary algorithms: A comparative study for evolutionary programming, evolutionary strategy, genetic algorithm, and linear programming. *IEEE Transactions on Power Systems*, 13(1), Feb 1998.

K.N. Miu, H.D. Chiang, and G. Darling. Capacitor placement, replacement and control in large-scale distribution systems by a GA-based two-stage algorithm. *IEEE Transactions on Power Systems*, 12(3):1160–1165, 1997.

A.J. Urdaneta, J.F. Gomez, E. Sorrentino, L. Flores, and R. Diaz. A hybird genetic algorithm for optimal reactive power planning based upon successive linear programming. *IEEE Transactions on Power Systems*, 14(4): 1292–1298, 1999.

D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

J.T. Ma and L.L. Lai. A new genetic algorithm for optimal reactive power dispatch. *Engineering Inteligent Systems for Electrical Engineering and Communications*, 5(2): 115–120, June 1997.

D.T.Y. Cheng and J.F. Macqueen. Some implimentation issues of a linear based reactive power optimisation program. In *12th Power Systems Computation Conference*, August 1996.

G. Levitin, S. Mazal-Tov, and B. Reshef. Genetic algorithms for optimal planning of transmission system reactive power compensation under security constraints. In *9th Mediterranean Electrotechnical Conference*, pages 897–900, 1998.

K.Y. Lee, Y.M. Park, and J.L. Ortiz. A united approach to optimal real and reactive power dispatch. *IEEE Transactions on Power Apparatus and Systems*, PAS-104:1147–1153, 1985.

Appendix B

The Beta study

The following tables detail the test study used in this work. The study has been developed, with cooperation from National Grid, to represent a practical test system. Table B.1 provides a summary of the system.

Table B.1: A summary of the test system

Busbars	68
Lines	141
Generators (LV)	9 (7)
Preventive transformer taps	5
FC candidate busbars	55
expandable SVCs	5

Note that the FC candidate busbars (55 of them), the expandable SVCs (5 of them), the LV generator busbars (7 of them) and the slack node (node XKX4) account for all of the 68 busbars.

The sizes of SVC that may be installed are given in table B.2, and the fixed compensation candidate nodes (FC nodes) may have capacitive compensation added in chunks of 60MVAr.

Table B.2: Available sizes of commercial reactive power compensation devices (National Grid, 2001)

System voltage (In kVs)	Type	Size of single device (In MVAr)	
		Inductive	Capacitive
275	SVC	106	150
400	SVC	75	150

The following table, B.3, details the contingencies applying to each of the five cases considered.

Table B.3: Contingencies

Case	Contingency
1	Intact: no contingency
2	Line out: A514
3	Line out: A42A
4	Line out: B513
5	Line out: F508

B.1 The Beta System

B.1.1 Lines

The following table details the lines of the Beta system.

Type	Bus1	Bus2	R	X	B	Sr
	XAX1				57.5	
4	XAX1	XAX2	0.19	8.3	-0.251	240
4	XAX1	XAX2	0.19	8.3	-0.251	240
4	XAX1	XAX2	0.15	8.5	-0.251	240
	XAX2	XGX2A	0.084	0.7501	5.1863	
2	XAX2	XLX2	0.1398	1.4273	10.3276	955
2	XAX2	XMX2	0.0948	0.9519	10.8477	760
	XBX1				38.27	
4	XBX1	XBX2	0.16	8.05	-0.251	240
4	XBX1	XBX2	0.16	8.02	-0.251	240
4	XBX1	XBX2	0.16	8.11	-0.251	240
2	XBX2	XLX2	0.0918	0.9123	6.5204	955
2	XBX2	XPX2	0.1228	1.1104	7.5762	955
2	XBX2	XPX2	0.1093	0.9886	6.7456	935
2	XBX2	XVX2	0.1209	1.2094	9.9255	42
	XCX2	XTX2A	0.0499	0.4956	49.5642	
2	XCX2	XVX2	0.0984	0.8915	7.4393	55
	XDX1				132.05	
4	XDX1	XDX2	0.17	8.2	-0.251	240
4	XDX1	XDX2	0.16	8.1	-0.251	240
4	XDX1	XDX2	0.16	8.2	-0.251	240
2	XDX2	XSX2	0.0209	0.2849	83.889	929
	XFX1				71.52	
	XFX1	XFX4	0.156	8.45	-0.317	240
	XFX1	XFX4	0.16	8.4	-0.317	240
	XFX1	XFX4	0.17	8.3	-0.317	240
1	XFX4	XJX4	0.0934	1.0215	33.6287	10
1	XFX4	XJX4	0.0934	1.0215	33.6287	10
1	XFX4	XKX4	0.0776	0.8485	27.9325	10
1	XFX4	XKX4	0.0777	0.8499	27.9766	10
	XGX1				13.2	
	XGX1	XGX2	0.1331	8.1466	-0.251	240
	XGX1	XGX2	0.15	8.6	-0.251	240
	XGX1	XGX2	0.15	8.5	-0.251	240
	XGX2	XGX2A	0.0005	0.001	0.1	
2	XGX2	XMX2	0.0464	0.7079	6.4601	130
2	XGX2	XZX2J	0.2122	2.4475	19.1426	795
1	XKX4	XMX4	0.0503	0.5499	18.1008	10
1	XKX4	XXX4T	0.0354	0.3873	12.7481	10
1	XKX4	XZX4	0.0289	0.3165	10.4189	10
3	XLX2	XLX4	0.0176	1.609	-0.381	120
4	XLX2	XLX6	0.31	15.8	-0.715	120
4	XLX2	XLX6	0.4583	16.616	-0.715	120
4	XLX2	XLX6	0.32	15.8	-0.715	120
1	XLX4	XMX4	0.0737	0.806	26.5315	10
1	XLX4	XNX4	0.1128	1.2342	40.6291	10
1	XLX4	XRX4	0.1851	2.01	64.9	0
1	XLX4	XHX4	0.2261	2.806	135.18	0
4	XLX6				8.02	
	XMX1J				8.2	

Type	Bus1	Bus2	R	X	B	Sr
	XMX1J	XMX2	0.38	11.6	-0.715	120
	XMX1J	XMX2	0.38	11.6	-0.715	120
	XMX1J	XMX2	0.38	11.6	-0.715	120
	XMX1K	XMX4	0.153	8.41	-0.317	240
	XMX1K	XMX4	0.153	8.42	-0.317	240
2	XMX2	XSX2	0.024	0.3668	3.3468	130
2	XMX2	XTX2	0.1438	1.36	12.1958	760
3	XMX2Q	XMX4	0.017	1.6	-0.463	0
2	XMX2Q	XSX2	0.0238	0.3631	3.3135	525
	XNX1				2.99	
4	XNX1	XNX4	0.16	8.4	-0.317	240
4	XNX1	XNX4	0.16	8.4	-0.317	240
1	XNX4	XQX4	0.1058	1.1574	38.1016	10
1	XNX4	XQX4	0.1058	1.1574	38.1016	10
1	XNX4	XVX4Q	0.052	0.4755	14.563	390
1	XNX4	XVX4R	0.0584	0.534	16.3523	390
1	XNX4	XXX4T	0.1268	1.1589	35.4911	390
	XPX1				135.93	
4	XPX1	XPX2	0.16	8.2	-0.251	240
4	XPX1	XPX2	0.16	8.2	-0.251	240
4	XPX1	XPX2	0.16	8.1	-0.251	240
2	XPX2	XTX2	0.0608	0.5523	6.139	760
2	XPX2	XUX2	0.0357	0.3307	8.6318	760
	XQX4	XQX4R		0.35		0
	XQX4	XQX4T		0.35		0
	XSX1				110.57	
4	XSX1	XSX2	0.15	8.6	-0.251	240
4	XSX1	XSX2	0.15	8.4	-0.251	240
4	XSX1	XSX2	0.14	8.5	-0.251	240
	XTX1				10.48	
4	XTX1	XTX2	0.1842	8.0888	-0.345	180
4	XTX1	XTX2	0.1842	8.166	-0.345	180
	XTX2	XTX2A	0.0005	0.001	0.1	
2	XTX2	XYX2Q	0.0194	0.1773	4.195	952
	XUX1				16.88	
4	XUX1	XUX2	0.19	8.7	-0.251	240
4	XUX1	XUX2	0.19	8.8	-0.251	240
1	XUX4	XUX4Q	0.0943	0.9254	37.21	0
3	XUX2	XUX4Q	0.017	1.6	-0.463	0
3	XVX2	XVX4Q	0.017	1.6	-0.463	150
3	XVX2	XVX4R	0.017	1.6	-0.463	150
	XVX1				28.4	
4	XVX1	XVX2	0.1543	8.5875	-0.251	40
4	XVX1	XVX2	0.15	8.6	-0.251	40
4	XVX1	XVX2	0.18	8.7	-0.251	40
	XXX1				19	
	XXX1	XXX4T	0.153	8.3333	-0.317	240
	XXX1	XXX4T	0.153	8.3333	-0.317	240
1	XXX4	XXX4T	0.0005	0.001	0.1	
	XYX1				8	
	XYX1	XYX2Q	0.1512	8.8333	-0.251	240
	XZX1				50.464	
9	XZX1	XZX2J	0.2944	12.283	-0.715	120
9	XZX1	XZX2J	0.2944	12.283	-0.715	120
9	XZX1	XZX2K	0.2981	12.25	-0.715	120
	XZX1	XZX4	0.153	7.92	-0.317	240
3	XZX2J	XZX4	0.0329	2.4153	5.6276	0
3	XZX2K	XZX4	0.0329	2.4148	5.4211	0
	XDX1				20	
1	XZX4				200	
1	XWX4	XZX4	0.0351	0.398	13.2053	10
1	XKX4	XWX4	0.056	0.6126	27.2691	10
	XEX2	XWX2S	0.019	0.2824	7.35	0
2	XAX2	XEX2	0.304	3.4589	27.16	95
	XEX1				4.26	
4	XEX1	XEX2	0.3039	12.316	-0.215	20
4	XEX1	XEX2	0.2875	12.25	-0.215	20
3	XWX2S	XWX4	0.017	1.6	-0.463	0
2	XCX2	XKX2K	0.1682	1.6062	11.2668	55
	XCX1	XYX1	0.277	0.568		
	XCX1				24.04	
4	XCX1	XCX2	0.1562	7.575	-0.251	40
4	XCX1	XCX2	0.153	7.75	-0.251	40
4	XCX1	XCX2	0.1602	7.6041	-0.251	40
2	XDX2	XKX2J	0.0796	1.2154	11.0907	40
2	XDX2	XKX2J	0.0796	1.2138	11.0766	40
	XKX1				2.4	
	XKX1	XKX2J	0.3777	11.933	-0.715	20
	XKX1	XKX2J	0.35	12	-0.715	20
	XKX1	XKX2K	0.3013	13.333	-0.715	20
	XKX2J	XKX2S		3.333		
3	XKX2J	XKX4	0.0213	1.6039	1.15	0

Type	Bus1	Bus2	R	X	B	Sr
3	XKX2J	XKX4	0.0213	1.6039	1.15	50
3	XKX2K	XKX4	0.0217	1.6084	3.1336	50
	XNX4	XNX81	0.025	1.285		
	XQX4	XQX81	0.01	0.9767		
	XWX4	XWX81	0.025	1.225		
	XWX4	XWX82	0.02	1.98		
	XJX4	XJX81	0.0167	0.8167		
	XJX4	XJX82	0.01	0.99		
	XHX4	XHX81	0.025	1.225		

B.1.2 Busbars

The following table details the busbars of the Beta system. The table also indicates if the busbar is a candidate for compensation siting: the "State" column indicates whether the busbar is a candidate for the installation of Fixed Capacitors (FC), Static VAr Compensators (SVC) or is the Slack bus (SLACK). Note that the the busbar voltage may be used in conjunction with table B.2 to evaluate the unit size of SVCs, where applicable.

The column labelled VSet is to be interpreted as follows: a one (1) indicates this is the target voltage for a corrective tap; the three letters OPT indicate this value requires optimisation as part of the planning exercise. If a busbar has a VSet value requiring optimisation and a value given in the Slope column, the VSet value is the fixed point of an SVC characteristic, otherwise it is the target voltage of a preventive tap.

Busbar Name	V (kV)	PG (MW)	QG/QGmin (MVAr)	QGmax (MVAr)	PL (MW)	QL (MVAr)	VSet (pu)	Slope (MVAr/V)	State
XKX4	400								SLACK
XVX2	275								FC
XCX1	132		120		234.155	98.287	1		FC
XCX2	275								FC
XEX1	132				124.304	47.216	1		FC
XEX2	275								FC
XKX1	132				159.958	60.032	1		FC
XVX4Q	400								FC
XVX1	132				300.643	134.904	1		FC
XKX2J	275								FC
XKX2K	275								FC
XKX2S	275		-75	150			OPT	-3000	SVC
XVX4R	400								FC
XXX1	132				185.405	78.178	1		FC
XYX1	132				117.028	49.348			FC
XYX2Q	275								FC
XZX1	132	207.7	107.7		443.532	198.196	1		FC
XZX2J	275								FC
XZX2K	275								FC
XZX4	400		-75	150			OPT	-3000	SVC
XBX1	132		180		445.153	200.699	1		FC

Busbar Name	V (kV)	PG (MW)	QG/QGmin (MVAr)	QGmax (MVAr)	PL (MW)	QL (MVAr)	VSet (pu)	Slope (MVAr/V)	State
XBX2	275								FC
XPX1	132				378.696	123.223	1		FC
XPX2	275								FC
XTX1	132				127.851	64.865	1		FC
XTX2	275								FC
XTX2A	275								FC
XUX1	132				107.116	51.352	1		FC
XUX2	275								FC
XAX1	132		180		323.066	126.326	1		FC
XAX2	275								FC
XDX1	132		20		384.666	125.123	1		FC
XDX2	275								FC
XGX1	132		180		500.681	195.794	1		FC
XGX2	275								FC
XGX2A	275		0						FC
XLX2	275		-75	150			OPT	-3000	SVC
XLX6	66				206.239	86.886	1		FC
XMX1J	132				216.456	91.291	1		FC
XMX1K	132				224.041	72.873	1		FC
XMX2	275								FC
XMX2Q	275								FC
XSX1	132				433.216	207.605	1		FC
XSX2	275								FC
XLX4	400		-75	150			OPT	-3000	SVC
XMX4	400								FC
XFX1	132				716.127	309.006	1		FC
XFX4	400		-150	300			OPT	-6000	SVC
XNX1	132				306.477	126.627	1		FC
XNX4	400						OPT		FC
XXX4	400	970	-300	469			1		FC
XXX4T	400								FC
XQX4	400						OPT		FC
XQX4R	400								FC
XQX4T	400								FC
XJX4	400						OPT		FC
XUX4Q	400								FC
XRX4	400				893	-98			FC
XHX4	400		0		1907	17.5	OPT		FC
XWX4	400						OPT		FC
XWX2S	275								FC
XNX81	Low	0	-160	620	30				
XQX81	Low	0	-285	930	90				
XWX81	Low	0	-150	620	34				
XWX82	Low	660	-185	410	35				
XJX81	Low	1506	-225	930	51				
XJX82	Low	1320	-370	820	70				
XHX81	Low	1004	-150	620	34				