Optimisation of Urban Water Supply Headworks
Systems Using Probabilistic Search Methods and
Parallel Computing

Lijie Cui (B.Sc, M.Sc)

A thesis submitted for the degree of Doctor of Philosophy at The
University of Newcastle

March 2003
DECLARATION

I hereby certify that the work embodied in this thesis is the results of original research and has not been submitted for a higher degree to any other University or Institution.

Lijie Cui

Lijie Cui
ACKNOWLEDGMENTS

This project would have been impossible to complete without the help of lots of people over the past four years. It is a real pleasure to acknowledge them here.

I wish to express my deep gratitude and sincere thanks to my supervisor, George Kuczera, for his invaluable guidance and constant support through the tortuous route to completion.

I would like to gratefully acknowledge the financial support provided by The University of Newcastle for my PhD scholarship.

The Sydney Catchment Authority is acknowledged for providing the data for the research.

My special thanks are owed to all the friends and colleagues at the Department of Civil, Surveying, and Environmental Engineering. I would also like to thank the computer support personal, Josh Kautto, for his assistance, Kim Vu for sharing the similar research experiences with me right through the end.

I am eternally grateful for the continual love and support provided by my family. My parents, who overcame a lot of difficulties, spent one year with me here to look after my kids; my sisters, who always gave me just that extra bit of encouragement when I needed it and always told me it was possible. Thank you all for always encouraging and believing in me, I am grateful forever.

Last but not least, I am indebted to my husband Daichao, my Don and Jasmine, for their patience and understanding for the many lost evenings and weekends during the time of writing this thesis. Particularly, I am indebted to my little Jasmine, who is so used to sleeping on the floor in my study room while I was writing at night, thank you for putting up with me! I affectionately dedicate this thesis to all my beloved ones.
Table of Contents

Lists of Tables
Lists of Figures

Abstract

Chapter 1  Introduction  1-1
  1.1 Efficient management of urban water supply – the need for optimisation  1-1
  1.2 Objectives……………………………………………………………………1-3
  1.3 Outline of the thesis…………………………………………………………1-6

Chapter 2  Review of optimisation methods in water resources systems……………………………………………………………2-1
  2.1 Introduction…………………………………………………………………2-2
  2.2 Traditional optimisation methods…………………………………………2-3
    2.2.1 Linear programming (LP)……………………………………………2-4
    2.2.2 Dynamic programming (DP)…………………………………………2-6
    2.2.3 Nonlinear programming………………………………………………2-9
    2.2.4 Concluding remark on traditional methods………………………..2-10
  2.3 Modern probabilistic search methods……………………………………2-11
    2.3.1 Genetic algorithms (GA)………………………………………………2-11
    2.3.2 Shuffled complex evolution method…………………………………2-17
    2.3.3 Simulated annealing (SA)……………………………………………..2-20
  2.4 Summary……………………………………………………………………2-23

Chapter 3  Genetic algorithms and the shuffled
Chapter 3  Applications of evolutionary approaches

3.1 Introduction

3.2 Genetic algorithms

3.2.1 A brief history of genetic algorithms

3.2.2 How GAs work?

3.2.3 Encoding

3.2.4 Selection methods

3.2.5 GA operators

3.2.6 Elitism

3.2.7 Assignment of GA parameters

3.2.8 Formulation of a customised GA

3.3 The shuffled complex evolution method

3.4 Summary

Chapter 4  Assessment of optimisation methods using a simple case study employing Monte Carlo simulation

4.1 Introduction

4.2 Urban water supply headworks system case study

4.2.1 Decision variables

4.2.2 Monte Carlo simulation of an urban water supply system

4.2.3 Objective function

4.2.4 Investigation of objective function surface

4.3 Efficiency and robustness of probabilistic search strategies
Chapter 5  Approaches for reducing computational turnaround time  

5.1 Introduction  5-2  
5.2 The need for parallel computing  5-2  
5.3 Parallel computation tool  5-3  
5.3.1 What is PVM?  5-4  
5.3.2 PVM components  5-5  
5.3.3 PVM programming structure  5-6  
5.3.4 PVM load balancing  5-7  
5.4 Parallel GA implementation  5-8  
5.5 Replicate compression  5-10  
5.5.1 Rationale for replicate compression  5-10  
5.5.2 Identification of critical period  5-12  
5.5.3 The replicate compression method  5-13  
5.6 Concluding remark  5-17  

Chapter 6 Case study: Sydney headworks system  6-1  

6.1 Introduction  6-2  
6.2 Description of WATHNET simulation model  6-3  
6.2.1 WATHNET overview  6-3  

4.3.1 Comparison of the two search methods  4-9  
4.3.2 Effect of replicate length  4-11  
4.3.3 Effect of GA parameters on performance  4-12  
4.4 Discussion  4-16  
4.5 Summary  4-17  

Chapter 5  Approaches for reducing computational turnaround time  5-1
Chapter 6  Application of WATHNET to the Sydney headworks system

6.2.2 Network linear programs
6.2.3 Example of network operation
6.2.4 Description of the WATHNET program
6.2.5 Input data – streamflow generation
6.2.6 Input data – demand generation

6.3 Application of WATHNET to the Sydney headworks system

6.3.1 Overview of the Sydney headworks system
6.3.2 Review of current headworks operating policy
  6.3.2.1 Drawdown curves
  6.3.2.2 Restriction policies
  6.3.2.3 Shoalhaven pumping mark

6.3.3 Network schematic of Sydney headworks system

6.3.4 Scope for optimising existing operating rules

Chapter 7  Case study: formulation of the objective function

7.1 Introduction
7.2 Economic evaluation of water restriction losses
  7.2.1 Theory of consumer surplus
Chapter 7  Economic modeling: Objective function modeling and water restriction policies..........................7-1

7.2.2 Consumer surplus and water restrictions ..............................................7-4
7.2.3 Estimation of economic loss due to water restrictions ......................7-6
7.2.4 Modified cost of water restriction policies ........................................7-8

7.3 Objective function formulation .............................................................7-10
7.3.1 Decision variables ............................................................................7-10
7.3.2 Objective function ............................................................................7-11
  7.3.2.1 $C_R$ - Penalty of imposing water restrictions on outdoor water use ..........7-12
  7.3.2.2 $C_P$ - Cost of pumping from the Shoalhaven system ..................7-13
  7.3.2.3 $C_D$ - Cost of using desalination option ..................................7-15
  7.3.2.4 $C_C$ - Penalty for rationing commercial and indoor water use ..7-16
  7.3.2.5 $C_F$ - Penalty of system failure .............................................7-18

7.4 Summary ...............................................................................................7-19

Chapter 8  Case study: results and discussion .............................................8-1

8.1 Introduction ............................................................................................8-2
8.2 Data input ...............................................................................................8-2
  8.2.1 Parallel Genetic Algorithm Parameters ............................................8-3
  8.2.2 WATHNET Simulation Model Input Data ..........................................8-3
8.3 Computational efficiency of parallel computing ....................................8-3
  8.3.1 Efficiency of Parallel Computing Algorithm ......................................8-4
  8.3.2 Performance of parallel ga for optimising the sydney headworks system ......................................................................................8-5
8.4 Results for replicate compression ..........................................................8-7
  8.4.1 Results ............................................................................................8-7
  8.4.2 Discussion .......................................................................................8-10
8.5 Results for optimising eight decision variables..................................8-16
  8.5.1 Optimisation Results Based on 1000 Replicate Simulations............. 8-16
  8.5.2 Discussion ..................................................................................... 8-20
  8.5.3 Sensitivity of results to the number of replicates ........................... 8-28
8.6 Conclusion ......................................................................................... 8-34

Chapter 9 Conclusions and recommendations.................................9-1

9.1 Introduction ...................................................................................... 9-2
9.2 Summary and conclusions ............................................................... 9-2
  9.2.1 Recent advances in optimisation models ....................................... 9-2
  9.2.2 Modern probabilistic search methods ......................................... 9-2
  9.2.3 Robustness and efficiency of the GA and SCE methods ............... 9-3
  9.2.4 Implementation of parallel GA and replicate compression ........... 9-4
  9.2.5 Optimisation of operation of Sydney water supply headworks system ...................................................... 9-5
    9.2.5.1 The power of parallel computing ............................................. 9-5
    9.2.5.2 Application of replicate compression ..................................... 9-6
    9.2.5.3 Optimisation of the Sydney headworks system ...................... 9-6
9.3 Recommendations and future directions ......................................... 9-7
9.4 Concluding remark .......................................................................... 9-9

References ............................................................................................ R-1

Appendix 1 PVM terminology ................................................................. A-1
Appendix 2 PVM library routine ............................................................. A-2
Appendix 3 Average monthly rainfall .................................................... A-5
Appendix 4 Number of rain days per month .......................................... A-6
Lists of Tables

Table 4-1       Search method performance as a function of replicate length
Table 6-1       Arc parameters for network in Figure 6.4  
Table 6-2       Sydney headworks restriction policy
Table 6-3       Description of nodes appearing in the schematic shown in Figure 6.8
Table 6-4       Description of the function of each node
Table 6-5       Description of arcs used in WATHNET
Table 6-6       Breakdown of Demand in Sydney
Table 6-7       Estimate of garden water use as a percentage of total residential use
Table 6-8       Classification of Sydney water demand
Table 6-9       Average monthly per capita indoor and outdoor water consumption
Table 6-10      Regression analysis summary
Table 6-11      Population used for the demand model
Table 8-1       Performance of parallel computation for 150 objective function evaluations involving 1000 50-year replicates
Table 8-2       Performance of parallel GA for Sydney headworks system
Table 8-3       Near-optimal four restriction levels obtained with replicate compression
Table 8-4       Near-optimal four restriction levels obtained without replicate compression
Table 8-5       Comparison of water restriction expected costs
Table 8-6       Comparison of the computational turnaround time on the 10-node Athlon PVM cluster
Table 8-7       Eight decision variables obtained from the GA for different population scenario years
Table 8-8       Desalination results obtained from the GA
Table 8-9       Objective function component penalties for optimising eight decision variables
Table 8-10      Average demand for each population scenario year
Table 8-11      Indicative costs for each operating option
Table 8-12      Penalty results for 2000 replicates
Table 8-13      Penalty results for 10000 replicates
Table 8-14      Decision variables for each replicate run and population scenario year
Table 8-15      Total penalty value for each replicate run and population scenario year
Table 8-16      Standard error of penalty (STDEV) for each replicate run and population scenario year
Table 8-17      95% Confidence interval on the total penalty
## Lists of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Flow chart describing a simple genetic algorithm</td>
<td>3-6</td>
</tr>
<tr>
<td>3.2</td>
<td>Illustration of binary string</td>
<td>3-7</td>
</tr>
<tr>
<td>3.3</td>
<td>Illustration of genetic crossover (a) direct and (b) indirect</td>
<td>3-11</td>
</tr>
<tr>
<td>3.4</td>
<td>Illustration of mutation</td>
<td>3-12</td>
</tr>
<tr>
<td>3.5</td>
<td>Illustration of inversion</td>
<td>3-12</td>
</tr>
<tr>
<td>3.6</td>
<td>Illustration of the evolution steps taken by each complex</td>
<td>3-19</td>
</tr>
<tr>
<td>4.1</td>
<td>Demand restriction rules</td>
<td>4-4</td>
</tr>
<tr>
<td>4.2</td>
<td>Objective functions surface for the decision subspace $\alpha$ and $\text{Trig}$ with $C = 0.05$ for different number of replicates</td>
<td>4-8</td>
</tr>
<tr>
<td>4.3</td>
<td>Best objective function values obtained by GA and SCE over 20 runs</td>
<td>4-11</td>
</tr>
<tr>
<td>4.4</td>
<td>Effect of crossover and PSS</td>
<td>4-13</td>
</tr>
<tr>
<td>4.5</td>
<td>Best objective function values obtained by GA over 20 runs for different algorithmic parameter combinations</td>
<td>4-14</td>
</tr>
<tr>
<td>4.6</td>
<td>Best objective function values obtained by GA over 20 runs for different algorithmic parameter combinations</td>
<td>4-15</td>
</tr>
<tr>
<td>5.1</td>
<td>A Master-Slave parallel application of the genetic algorithm</td>
<td>5-9</td>
</tr>
<tr>
<td>5.2</td>
<td>Pseudo-code for Pool of Task load balancing</td>
<td>5-11</td>
</tr>
<tr>
<td>5.3</td>
<td>Schematic illustrating how restriction penalty is imposed</td>
<td>5-12</td>
</tr>
<tr>
<td>5.4</td>
<td>Example of critical periods: Mitta Mitta River</td>
<td>5-13</td>
</tr>
<tr>
<td>5.5</td>
<td>Pseudo code for replicate compression</td>
<td>5-16</td>
</tr>
<tr>
<td>5.6</td>
<td>Illustration of replicate compression using WATHNET</td>
<td>5-17</td>
</tr>
<tr>
<td>6.1</td>
<td>Schematic showing interaction between genetic algorithm and simulation model of the headworks system</td>
<td>6-2</td>
</tr>
<tr>
<td>6.2</td>
<td>Sample water supply headworks system</td>
<td>6-5</td>
</tr>
<tr>
<td>6.3</td>
<td>Graph of NetLP for sample water supply headworks system</td>
<td>6-6</td>
</tr>
<tr>
<td></td>
<td>shown in Figure 6.2</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>Example: demonstrating a demand shortfall due to water shortage</td>
<td>6-8</td>
</tr>
<tr>
<td>6.5</td>
<td>Schematic of WATHNET programs</td>
<td>6-11</td>
</tr>
</tbody>
</table>
Figure 6.6  Flowchart of generating seasonal streamflow data 6-13
Figure 6.7  Schematic of Sydney headworks system 6-15
Figure 6.8  Existing drawdown rules 6-18
Figure 6.9  Schematic of demand restriction rule 6-20
Figure 6.10  EDNET schematic of Sydney headworks system 6-22
Figure 6.11  Illustration of Warragamba pump mark in the WATHNET 6-26
Figure 7.1  A demand curve for a typical household 7-3
Figure 7.2  Household demand-price relationship showing loss of revenue and consumer surplus 7-5
Figure 7.3  Cost of restrictions when $q_2 \geq q_x$ 7-9
Figure 7.4  Cost of restrictions when $q_2 < q_x$ 7-10
Figure 7.5  Mapping between four restriction trigger levels and decision variables 7-14
Figure 8.1  Example of restriction rule set-up in SIMNET 8-11
Figure 8.2  Time series trace for the lowest, 4 and 99 percentile total system storage 8-11
Figure 8.3  System performance for population scenario year 2000: replicate 860 8-13
Figure 8.4  System performance for population scenario year 2020: replicate 860 8-14
Figure 8.5  System performance for population scenario year 2040: replicate 860 8-15
Figure 8.6  Near-optimal restriction trigger levels for each scenario year 8-18
Figure 8.7  Near-optimal pump marks for each scenario year 8-18
Figure 8.8  Percentage distribution of objective function component penalties for each population scenario year 8-21
Figure 8.9  System annual shortfall probability for year 2000 8-25
Figure 8.10  System annual shortfall probability for year 2040 8-25
Figure 8.11  Annual shortfall probability for outdoor water use 8-26
Figure 8.12  Annual shortfall probability for outdoor water use 8-26
Figure 8.13  Annual shortfall probability for commercial water use 8-27
Figure 8.14  Annual shortfall probability for commercial water use 8-27
Abstract

Realistic optimisation of the operation and planning of urban water supply headworks systems requires that the issues of complexity and stochastic forcing be addressed. The only reliable way of accomplishing this is to use simulation models in conjunction with the Monte Carlo method which generates multiple hydro-climate replicates. However, such models do not easily interface with traditional optimisation methods.

Probabilistic search methods such as the genetic algorithm (GA) and the shuffled complex evolution method (SCE) can be coupled to a generalised simulation model and thus accommodate complexity as well as stochastic inputs. However, optimisation of complex urban water supply systems is computationally intractable if Monte Carlo methods have to be used.

This study first compared the GA and the SCE method using a simple case study. Both methods were found to cope well with the piecewise flat objective function surface typical of the headworks optimisation problem. This is because they have the inherent capability of vigorously exploring beyond the domain of a flat region. The SCE method is recommended especially when fast location of a good solution is desired. Nonetheless, the GA was preferred due to its inherent parallelism.

Two methods were then explored to improve computational efficiency and turnaround time: parallel computing and replicate compression. The Sydney headworks system was used as a case study to investigate the key aspects of a full-scale headworks optimisation. It was concluded that the speedup was nearly proportional to the number of processors employed. Replicate compression can very significantly reduce the computational turnaround time for Monte Carlo simulation; unfortunately, this conclusion must be tempered by the limitation that the objective function depends on penalties arising from restrictions only. Critical analysis of the GA results suggested the optimised results were sound. The case study demonstrated the feasibility of parallel GA to identify near-optimal solutions for a complex system subject to stochastic forcing.
1.1 EFFICIENT MANAGEMENT OF URBAN WATER SUPPLY – THE NEED FOR OPTIMISATION

In the last two decades, there has been a significant shift of government policy in Australia from planning and construction of water resource projects to the efficient operation of existing infrastructure. The reasons include the non-availability of water resources for further development, the limited availability of funds for capital works and the spirited lobbying of environmental groups against construction of new dams or water storage projects. These changes have given new impetus to seeking optimal or near optimal solutions for managing urban water supply headworks systems (Perera and Codner 1998). This is the focus of this study.

A typical urban water supply system consists of a water supply headworks and a distribution system. The water supply headworks system comprises a network of reservoirs, possibly complex, connected by a transfer system which links the reservoirs. The distribution system distributes water within the supply area and generally consists of pipelines, pumping stations, channels and small reservoirs. This system meets the varying daily demands of industrial, domestic and other users. Whereas the distribution system is designed to meet peaks in daily demand, the headworks system is designed to provide adequate security against drought which may persist from months to years.

The objectives of operating a typical water supply headworks system usually reflect the need to achieve an acceptable level of reliability, to minimise operating costs and to defer the time to the next augmentation. The decision variables for operation and planning of headworks systems may include target levels for storages, target pumping levels and rules for the introduction of restrictions. Water restrictions of varying severity are imposed at times when it is deemed that the probability of failure (i.e. the system being unable to meet demand) is beyond a certain critical level. Major urban water supply authorities have policies indicating when water restrictions of varying severity will be applied. Currently these levels and rules are based on experience, supplemented in some cases by the use of computer simulation models (Dandy 1987). The application of optimisation algorithms to water supply
headworks systems offers the potential for more efficient management of existing systems. This may in turn result in potential benefits through delaying the need for major new capital works and through reductions in operating costs.

Headworks systems are characterised by their complexity and stochastic forcing. Typically they have multiple users, possibly with different objectives and risk tolerances, and also multiple sources with different levels of quality. In the last two decades, major advances in simulation technology have heralded the advent of generalised models which can be readily applied to a diverse range of complex systems. They can be used to describe the behaviour of the system in response to hydro-climatic inputs (streamflow, rainfall, etc.), anthropogenic requirements (urban, industrial and irrigation demand, etc.) and other environmental requirements given an infrastructure and an operating policy. In Australia, several models have gained considerable industry acceptance ([e. g., REALM based on the work of Kuczera and Diment 1987, IQQM 1995, HOMA Crawley and Dandy 1993, WATHNET Kuczera 1992, Salbe et al. 1994]. Because of the ease with which infrastructure options and operating policy options can be changed, these models have enabled comprehensive ‘what-if’ assessments.

When coupled with probability models of hydro-climatic inputs and anthropogenic requirements, generalised models can perform Monte Carlo simulation to produce a risk analysis of the consequences of droughts and other environmental stresses. It is widely accepted that simulation models using hydro-climatic inputs generated by Monte Carlo methods provide the most realistic models of future water resource system behaviour. Unfortunately, such models offer limited scope for optimising system performance, typically requiring a trial-and-error approach.

There are few realistic applications of optimisation techniques in water supply headworks systems. The technology used to plan and operate water supply headworks systems remains wanting in its ability to identify good or near-optimal solutions. Traditional optimisation approaches using linear, nonlinear, and dynamic programming have only been successfully applied to systems with known future
inputs and/or low levels of complexity. They have failed to adequately cope with the complexity and the stochastic behaviour typical of large water resource systems (Yeh 1985, Wurbs et al. 1985, Wurbs 1993).

If realistic optimisation of operating rules and infrastructure decisions is to be realised there appears little choice but to use Monte Carlo simulation. Unfortunately such models can be computationally very demanding and do not easily interface with traditional optimisation methods. The challenge, therefore, is to couple the Monte Carlo simulation with optimisation in a way that is computationally practical yet does not introduce restrictive assumptions about system behaviour.

1.2 OBJECTIVES

The primary goal of this thesis is to develop and demonstrate optimisation/simulation methodologies, which offer the promise of practical and realistic optimisation of the operation and management of urban water supply headworks. To this end, several specific objectives are pursued in this thesis:

1. To explore the use of modern probabilistic search strategies.

As already noted, simulation models in conjunction with the Monte Carlo approach do not easily interface with traditional optimisation methods. However, probabilistic search methods do not suffer from this limitation. There are a host of probabilistic search strategies such as genetic algorithms (Goldberg 1989), simulated annealing (Metropolis et al. 1953), and the shuffled complex evolution algorithm (Duan et al. 1992). Recent developments in probabilistic search methods have significantly improved their efficiency. Because they can be coupled to a generalised simulation model they can accommodate complexity as well as stochastic inputs. Moreover, they are typically robust and capable of converging to a global rather than local optimum. However, their ability to optimise water supply headworks systems evaluated using Monte Carlo simulation is not well known. This thesis aims to identify an appropriate
probabilistic search method that can be used to identify a near-optimal solution in large-scale complex water resources systems.

2. To explore the benefit of high performance computing to make optimisation practically possible.

Realistic optimisation of the operation and planning of urban water supply headworks systems requires that the issues of complexity and stochastic forcing be addressed. The only reliable way of accomplishing this is to use simulation models in conjunction with the Monte Carlo method which generates multiple hydro-climate replicates. To accurately estimate risk-based performance measures, an extremely large number of hydro-climatic replicates is usually required. As the complexity of a system increases the number of decision variables requiring optimisation will increase. Therefore, the computational time required to adequately evaluate the performance of a complex system can be substantial. In such cases, the computational efficiency of the Monte Carlo simulation and the search method is of critical importance.

Although modern probabilistic search methods are well suited to interfacing with Monte Carlo simulation models, there have been few reported studies involving the optimisation of large-scale complex headworks systems subject to stochastic forcing. Perhaps the main deterrent to such studies is the massive computational burden. To appreciate the magnitude of the computational effort, the optimisation of the headworks system for Sydney, Australia, is considered. The headworks system consists of a network of nine major reservoirs and an inter-basin transfer scheme to service a population of approximately 4 million. A Monte Carlo simulation involving 1000 replicates over a 50-year planning horizon takes about 310 CPU seconds on a 750 MHz Pentium III processor. A probabilistic search method optimising 5 to 10 decisions may require evaluation of the order of 1000 trial solutions using Monte Carlo simulation before converging to a near-optimal solution - this represents over 86 hours of CPU time! Clearly, such a task is impractical using PCs and remains in the domain of super or parallel computing.
Parallel computers can significantly improve the turn-round time for probabilistic search methods because the solutions of large complex problems can be conducted in parallel rather than sequentially. This study explores the benefits of parallel computing and the ability of probabilistic search methods to exploit parallelism.

3. To investigate methods to further improve the efficiency of the Monte Carlo simulation.

As said before, for urban water supply systems with high levels of reliability, Monte Carlo simulation can be computationally very intensive. This study, therefore, explores methods for improving the efficiency of Monte Carlo simulation of such high reliability systems.

4. To demonstrate the optimisation methodology using a realistic case study.

Because the overall objective is to develop an optimisation approach that is both practical and realistic it is essential that the proposed approach be demonstrated in an application involving a real and complex headworks system. In this study the Sydney headworks system was considered. It is a complex system containing nine reservoirs, multiple demand sites and pumped storage. The proposed optimisation approach is used to find near optimal pumping, desalination and restriction rules which minimise the total expected cost arising from pumping, water restrictions and desalination. In the context of the Sydney application, this study evaluates the strengths and weakness of the proposed approach.

1.3 OUTLINE OF THE THESIS

This thesis consists of nine chapters. Chapter 2 presents a review of current available optimisation approaches. Particular emphasis is given to modern probabilistic search methods. Two promising and widely used probabilistic search methods, the genetic algorithm (GA) and the shuffled complex evolution (SCE) algorithm, are then examined in Chapter 3. Their relative performance when coupled with Monte Carlo simulation is investigated using a simple hypothetical water supply
headworks system in Chapter 4. Optimisation of large urban headworks systems is so computationally demanding that parallel computing must be used. Chapter 5 introduces two approaches to reduce the computational burden. The first approach involves application of parallel computing techniques. The Parallel Virtual Machine protocol is used to implement GA optimisation in a parallel computing environment. The second approach implements replicate compression, which, to some extent, has potential to reduce Monte Carlo simulation effort for high-reliability multi-reservoir systems.

Chapter 6 introduces the Sydney headworks system as a case study to demonstrate the optimisation method. It then describes how to stochastically generate the streamflow and demand data required by the WATHNET simulation model. WATHNET, a software package currently used in the water industry for simulating headworks systems, is presented. Chapter 7 describes the decision variables and formulates the objective function used in the optimisation. Specifically an objective function based on pumping, and restriction costs is formulated along with constraints on system reliability. Chapter 8 presents and discusses the results obtained from optimising the Sydney headworks system. Chapter 9 summarises the main findings of this study and identifies future directions of research.
CHAPTER 2

REVIEW OF OPTIMISATION METHODS IN WATER RESOURCES SYSTEMS
2.1 INTRODUCTION

One of the most important advances in the field of water resources engineering during the last few decades has been the evolvement and application of optimisation techniques in planning, design, and management of complex water resources systems. There have been many successful applications in reservoir studies, mostly in the context of planning. Nonetheless, there is no general algorithm that comprehensively deals with the complex, nonlinear stochastic systems of water resource applications (Yeh 1985, Wurbs et al. 1985, Wurbs 1993). The choice of optimisation method depends on the characteristics of the system, the availability of data, and the objectives and constraints specified.

Generally speaking, most optimisation models are based on some type of mathematical programming technique. Traditionally, optimisation techniques have been divided into three distinct categories: (1) linear programming (LP), (2) dynamic programming (DP) and (3) nonlinear programming (NLP). In recent years, a new group of optimisation techniques based on probabilistic search concepts has appeared in the literature. Each of these techniques can be applied in a deterministic or stochastic environment and has a certain distinguishing feature that separates it from the others. Each has a certain type of problem for which it is best suited (Wurbs 1993). The modeller must be aware of the strengths and limitations of each model type to effectively solve the problem.

All optimisation methods are formulated in terms of determining values for a set of decision variables that will maximise or minimise an objective function subject to constraints. The objective function is the heart of an optimisation model. It is usually expressed mathematically as a function of the decision variables. Development of an objective function for evaluating the performance of alternative plans of operation is a crucial aspect of optimising complex reservoir operations. The success of the optimisation model depends on how closely the system response to alternative policies is reflected in the objective function. Difficulty in describing complex system objectives and performance criteria in the required mathematical format represents one of the most severe limitations encountered in applying traditional optimisation
techniques. The set of decision variables defines how the system is to be operated. The constraints on the optimisation model force it to obey physical laws, economic requirements, and social as well as other requirements (Simonovic 1992).

This chapter reviews traditional and more recent applications of optimisation in water resources engineering with the objective of identifying the most promising approaches to be applied to complex nonlinear stochastic systems. The emphasis of the review will be to identify the strengths and weaknesses of various approaches rather than detailed review of each individual approach.

### 2.2 TRADITIONAL OPTIMISATION METHODS

The literature describing optimisation models in general and applications to reservoir operations in particular is extensive. Loucks et al. (1981) described in detail mathematical programming models in water resources system analysis. Yeh (1985) presented a comprehensive in-depth state-of-the-art review of reservoir management and operation models, with a strong emphasis on optimisation methods. LP was covered in an elaborate manner. Various simulation models and real-time operation models used in water resources problems and their applications were also reviewed. Yakowitz (1982) reviewed exclusively DP models and discussed in detail the role and suitability of dynamic programming in reservoir operation, highlighting computational considerations. Wurbs et al (1985) extended the work of Yeh by producing a state-of-the-art review together with an annotated bibliography of system analysis techniques applied to reservoir operation. Esogbue (1989) presented an overview of various aspects of DP and problem areas of water resources, and a survey of water resources applications including reservoir optimisation. Simonovic (1992) presented a short review of mathematical models in reservoir management and operation. This work was aimed at presenting conclusions drawn by previous state-of-the-art reviews, and generating ideas for closing the gap between theory and practice. Wurbs (1993) reviewed reservoir system simulation and optimisation models. This review was done with the objective to contribute to ongoing efforts throughout the water management community in sorting through the numerous reservoir system-analysis models and better understanding which methods might be
most useful in various types of decision-support situations. Most of the models reviewed above had the form of linear or dynamic programming. Several of these models are cited in the following as a representative sampling of the variety of ways in which optimisation techniques have been applied.

2.2.1 Linear Programming (LP)

A linear program is an optimisation problem that can be expressed in the following so-called standard form:

\[
\text{minimize } \mathbf{c}^T \mathbf{x} \\
\text{subject to } \mathbf{A} \mathbf{x} = \mathbf{b} \\
\mathbf{x} \geq 0
\]  

(2.1)

where \( \mathbf{x} \) is the vector of decision variables to be optimised, the \( T \) refers to transpose, \( \mathbf{A} \) is a matrix of known coefficients, and \( \mathbf{c} \) and \( \mathbf{b} \) are vectors of known coefficients. The expression “\( \mathbf{c}^T \mathbf{x} \)” is called the objective function, and the equations “\( \mathbf{A} \mathbf{x} = \mathbf{b} \)” and “\( \mathbf{x} \geq 0 \)” are called the constraints. The matrix \( \mathbf{A} \) is generally not square, hence one cannot solve the LP by just inverting \( \mathbf{A} \). Usually \( \mathbf{A} \) has more columns than rows, and \( \mathbf{A} \mathbf{x} = \mathbf{b} \) is therefore quite likely to be under-determined, leaving considerable latitude in the choice of \( \mathbf{x} \) with which to minimise \( \mathbf{c}^T \mathbf{x} \).

LP has been judged to be one of the most widely used optimisation techniques in water resources and one of the most important scientific advances in recent history (Simonovic 1992). The first application in reservoir operation dates back to Dorfman (1962), who applied LP to a simplified reservoir problem without over-year storage. Windsor (1973) developed a methodology using a recursive LP as the optimisation tool for the analysis of a multi-reservoir flood control system. The objective function was to synchronise the reservoir release schedules so as to minimise the total damage cost at all locations being protected, given the updated runoff hydrographs at all significant points in the system. Loucks et al (1981) have presented a number of LP reservoir problem formulations for a deterministic problem based on maximising
reservoir yield. Randall et al (1990) developed a LP model to study the operation of a metropolitan water system during drought consisting of multiple reservoirs, groundwater, treatment plants, and distribution facilities. Four objectives were incorporated in the modelling process. Alternative versions of the model were formulated with one objective being optimised as the objective function, with the other objectives being incorporated as constraints with user-specified levels. Trade-off curves were developed to show the trade-offs between the four alternative objectives.

LP provides a simple, efficient tool to assess operational policies and basic system reliability. There are some important applications along with additional techniques that extend and amplify the usefulness of LP (Simonovic 1992). Linear programming has been extensively used in stochastic reservoir modelling as the main technique within different approaches such as: chance constrained LP, stochastic LP for Markov process, stochastic programming with recourse, and reliability programming. LP can be used to obtain preliminary assessments of operational policies for planning and management purposes. The advantages of LP include:

(1) It can accommodate relatively high dimensionality with comparative ease.

(2) The global optimal solution can be obtained.

(3) No explicit initial solution is needed.

(4) Standard computer codes are readily available (Yeh 1985).

However, LP is limited to solving only linear problems, i.e. problems with objective function and constraints in linear forms. LP can be computationally expensive in solving multiperiod optimisation problems for large complex reservoir problems with large numbers of decision variables and constraints (Yeh 1985, Sun et al. 1995). Although there are techniques which allow some nonlinear problems to be solved by linear programming, for example, the introduction of integer variables when adapting LP to nonlinear problems, these techniques introduce approximations of the original problem and tend to increase the number of constraints on the
problem, which in turn, leads to computationally more expensive solutions. The dimensionality problem associated with the stochastic LP approach remains unsolved.

2.2.2 Dynamic Programming (DP)

Dynamic programming is an efficient mathematical approach for making a sequence of interrelated decisions. It is very well suited to situations involving multistage decision problems such as reservoir operations (Wurbs 1993). The popularity and success of this technique is attributed to the fact that the nonlinear and stochastic features, which characterise a large number of water resources systems, can be translated into a DP formulation. The technique involved is not restricted to any particular problem structure. Unlike LP, development of a generalised algorithm and corresponding computer program for all DP problems is impossible. Detailed reviews of DP have been provided by Wurbs et al (1985), Yeh (1985), and Yakowitz (1982). There have been two classes of DP approaches used in reservoir optimisation in the past when the stochasticity of streamflows was included: stochastic dynamic programming (SDP) and deterministic dynamic programming (DDP) (Codner 1979, Yakowitz 1982).

The DDP approach uses multiple sequences of synthetic streamflow data to obtain multiple release patterns, which are then combined together to obtain a single mean release rule. Hall and Howell (1963) suggested the use of deterministic, synthetically generated streamflow sequences, hence the name of deterministic dynamic programming. Young (1967) applied the approach of Hall and Howell (1963) to a single reservoir problem to find the mean release policy on an annual basis. McKerchar (1971) used a similar approach to Young (1967) on two examples using New Zealand catchments. He concluded that the deterministic algorithm is more efficient in terms of computational requirements than the stochastic dynamic approach. Codner (1979) adopted DDP in conjunction with multiple sequences of generated streamflow for an urban water supply headworks system. The implicit stochastic optimisation (ISO) method of Croley (1974) is another example of DDP under a different heading. Multiple flow sequences were used to produce optimal
release from which a mean release policy was derived by means of regression analysis.

The advantage of DDP is that it allows for the stochastic nature of the streamflow without incurring the dimensionality problems encountered with SDP. The drawback with this approach is that the operating rules tend to be optimistic because they are developed with perfect knowledge of future inflows. This knowledge is not available in actual operation. A further limitation is that a maximum number of three or four storages can be handled in a practical sense (Dandy 1987). Another drawback is that the method relies heavily on the regression analysis (Karamouz and Houck 1982) and the mean operating rules do not represent the true optimal release (Saad and Turgeon 1988). There have been subsequent developments (Karamouz and Houck 1982, 1987, Karamouz et al. 1992) attempting to overcome the above drawback by refining the operating rules through regression equations. Nonetheless, the mean operating rules do not represent the true optimal operating rules.

The SDP approach employs a statistical description of the streamflow and forecast process instead of a special streamflow sequence to obtain optimal policies. It considers the stochasticity of streamflow explicitly through streamflow distributions. It does not consider multiple streamflow sequences sampled from these distributions. Therefore, it eliminates the need for combining the optimal operating rules from each sequence into mean optimal operating rules. Storage volume is discretized into a number of storage states. The approach then generates a single operating policy for every possible storage and inflow state for each time interval. Therefore, SDP is potentially the best method that can be used to derive the optimal operating rules for multiple reservoir systems (Codner 1979).

The earliest reservoir operation study using SDP in the English language appears to be the work of Little (1955). He determined an optimal operating policy for a single storage supplying a hydroelectric power station. Inflows were assumed to be independent variables. Buras (1965) applied the stochastic method to a conjunctive surface and groundwater storage system in order to find the optimum operation policy. Askew (1973) applied stochastic dynamic programming in similar ways to
the single storage system. The combination of the DP and simulation models made this approach highly costly in terms of computation time. Stedinger et al (1984) developed a SDP model, which employed the best forecast of the current period’s inflow to define a reservoir release policy and to calculate the expected benefits from future operations. Substantial improvements in simulated reservoir operations with derived stationary reservoir operating policies were reported, but whether this technique provides better operating policies in real-time operation was not established.

Applications of SDP to reservoir problems have been reported by numerous authors. However, the stochastic methods can only handle a limited number of storages and cannot be applied to large-scale systems (Philbrick and Kitanidid 1999). The usefulness of SDP is limited by its prohibitive computational requirement for problems with more than two or three state variables, such as the case of the operation of multiple reservoir systems (Turgeon 1980, Loucks and Sigvaldason 1982, Yakowitz 1982, Yeh 1985, Sharif and Wadlaw 2000).

Over the years, a number of attempts have been developed to alleviate the problem of dimensionality associated with finding the optimal solution in large-scale stochastic models, and make it more favourable to solve certain specific problem. These attempts reduce either computation time or computer storage requirements, or increase the applicability of the solution technique. Larson (1968) obtained the solution to the four-reservoir problem using incremental dynamic programming (IDP). In fact, this four-reservoir problem has become somewhat of a benchmark test problem. Similar work had been done by Hall et al (1969) and Heidari et al (1971) using discrete differential dynamic programming (DDDP). DDDP can be viewed as different version of IDP, and the two terms have been used interchangeably in water resources applications (Sharif and Wardlaw 2000). IDP overcomes the dimensionality problem to a large extent, but requires stringent conditions to be satisfied for achieving an optimal solution. Another approach that overcomes the dimensionality problem is the DP successive approximation technique (Larson 1968, Trott and Yeh 1973, Nompngcol and Askew 1976). However, its common
drawback is convergence to a local optimum. Murry and Yakowitz (1979) have used a four-reservoir configuration to illustrate the efficiency of Constrained Differential Dynamic Programming. Application of the method to a ten-reservoir problem has demonstrated its ability, which is thought to be beyond the capability of other methods. The main feature of this method is the limited amount of computer storage required. However, the technique requires that the objective function be differentiable and the constraints linear. Trezos and Yeh (1987) extended the use of differential DP to a reservoir management problem. It can be applied to large-scale systems without discretizing the state and control variables under the limitation that the recursive equation is a concave function of the state variables. Optimal control theory (OCT) also provides an efficient means of overcoming the dimensionality problem. The requirement that the objective function must be differentiable limits the application of the approach to some extent. There have been other advances in this area including the work done by Stedinger et al. (1985), Wang and Adams (1986), Karamouz and Houck (1987), Mizyed et al. (1992), and others.

In brief, DP is capable of coping with nonlinear and stochastic problems. It is specifically suitable to those reservoir planning and operation problems which can be represented as either a progressive or serial directed network problem. However, DP can only be applied to less complex systems due to the dimensionality problem. The nonseparable objective function required in the DP model also limits its applicability. None of the studies found in the literature can deal simultaneously with the following aspects of the planning and operation problems: nonlinearity, stochastic streamflows, and complexity (multiple reservoirs, multiple periods).

### 2.2.3 Nonlinear Programming

Mathematically, NLP can be expressed as follows:

$$\text{Min } f(x)$$

subject to  \( g(x) = b \)

and  \( v \leq x \leq u \)  \( \text{(2.2)} \)
where the vectors $v$ and $u$ of the inequality constraints represent lower and upper bounds for decision vector $x$.

NLP offers a more general mathematical formulation of the reservoir problem. NLP techniques include search techniques, quadratic programming, geometric programming, and separable programming which can be used iteratively in large-scale system problems. They can be used in conjunction with simulation as well as other programming techniques. NLP can effectively handle a nonseparable objective function and nonlinear constraints which many other programming technique cannot. Search techniques have been also applied in conjunction with simulation to evaluate the system performance. The application of NLP has been reported by Hicks et al. (1974), Haimes (1977), Rosenthal (1980), Simonovic and Marino (1980) and others.

Nonlinear programming (NLP) has been treated extensively in the literature of operation research. However, NLP has not enjoyed the popularity that LP and DP has in water resources systems analysis. The main reason is that NLP techniques are slow, iterative and take up large amounts of computer storage and time. Most techniques for nonlinear optimisation only find a local optimum. Moreover, NLP, unlike DP, cannot easily interface with the stochastic nature of streamflows to the system. Overall, NLP is considered not viable in applications involving complex and stochastic systems.

### 2.2.4 Concluding Remark on Traditional Methods

Although traditional optimisation methods including linear, nonlinear and dynamic programming can provide efficient computation procedures for achieving an optimal solution to a range of problems, these approaches become computationally bounded very rapidly (Esat and Hall 1994). Overall, the methods suffer from several disadvantages:

1. They may be ineffective at finding the optimal solution. There is no guarantee that the obtained solutions are the best ones.
2. Imposing a separability condition on the objective function limits its applicability.
3. Because of the problem of dimensionality, the research in the area of stochastic optimisation is usually limited to small scale problems.

In conclusion, the most significant finding is that none of the studies found in the literature are able to deal comprehensively with the following aspects of the planning and operation problems - nonlinearity, input stochasticity, and complexity (multiple reservoirs, multiple periods).

2.3 MODERN PROBABILISTIC SEARCH METHODS

During the last two decades, probabilistic search methods have emerged as valuable optimisation tools capable of solving global optimisation problems in many hydroinformatic applications, such as calibration of hydrological models, optimal control of hydrosystems, water quality management, water supply distribution and sewage network design, etc. These global optimisation algorithms involve the evaluation of the objective function usually at a random sample of points in the feasible parameter space, followed by subsequent manipulations of the sample using a combination of deterministic and probabilistic rules. They can have a high probability of locating the near-optimal solution. In this section, some of the popular global probabilistic search methods will be reviewed, such as genetic algorithms, simulated annealing, and the shuffled complex evolution method.

2.3.1 Genetic Algorithms (GA)

The genetic algorithm is a search procedure for finding the minimum or maximum of an unconstrained function using random selection processes, e.g. based on heuristic analogues of Darwinian natural selection and genetic reproduction. There is no single algorithm but rather a range of heuristic variations of a general algorithm adapted to suit the particular problem. In recent years, a number of researchers have applied the genetic algorithm technique to water resources problems.

- Goldberg and Kuo (1987) were the first to apply GAs to a water resource problem, which involved an optimisation problem in a pipeline. Since then, a
number of applications of GAs to pipe network problems have been reported in
the literature:

Murphy et al (1993) developed a methodology for optimising a water supply
network using a simple GA. The objective was to find the combination of pipe
sizes that minimised the cost of the water distribution network.

Simpson et al (1994) applied the GA technique to a case study pipe network and
compared the GA performance against the classical techniques of complete
enumeration and nonlinear programming (NLP). They concluded that the GA
was effective in finding global optimum or near-optimum solutions for the case
study, although it was not as fast as NLP.

Davidson and Goulter (1995) used GAs to optimise the layout of a branched
rectilinear network, such as a natural gas or water distribution system.

Dandy et al (1996) developed an improved GA for the New York City water
distribution problem. It was reported that the solution found was the lowest cost
design yet presented in the literature for that particular problem. There were
three main changes to the simple GA that significantly enhanced its
performance: (1) variable power scaling of the fitness function, (2) an adjacency
or creeping mutation operator, and (3) use of Gray codes to represent the set of
decision variables.

Savic and Walters (1997) developed the computer model GANET for the least-
cost design of water distribution networks, and again demonstrated the potential
of GANET as a tool for water distribution network planning and management.

Reis et al (1997) reported an application of a simple GA to the problem of
optimal location of control valves in a water supply network with leakage
reduction as the objective function to be minimised. The application showed
reasonably good solutions involving a small number of valves.

Halhal et al. (1997) described a multi-objective optimisation approach using
capital cost and benefit as dual objectives to the network rehabilitation. A
structured messy genetic algorithm was developed, incorporating some of the principles of the messy genetic algorithm, such as strings that increase in length during the evolution of designs. The algorithm was shown an effective tool for that optimisation problem and the algorithm performed much better than a standard genetic algorithm for a large network.

Montesinos et al (1999) proposed a modified genetic algorithm for water distribution network optimisation. They obtained the lowest-cost feasible solution reported in the literature in far fewer generations than any previous GA.

Meier and Barkdoll (2000) applied a GA to a network analysis model for a small town in Ohio and validated against the optimal solutions found by complete enumeration in a series of validation tests.

Wu and Simpson (2001) applied a GA to the optimal design and rehabilitation of a water distribution system. This work employed a messy genetic algorithm to enhance the efficiency of an optimisation scheme in a large-scale water distribution system problem. Two benchmark problems of water pipeline design and a real water distribution system were presented to demonstrate the improved technique. The results showed that the number of design trials required by the messy genetic algorithm is consistently fewer than used by other genetic algorithms.

- Genetic algorithms have also successfully been used as an optimisation tool in water quality and runoff modelling.

Wang (1991) applied a GA to the calibration of a conceptual rainfall-runoff model. The model had seven calibration parameters; the values of which were optimised by minimising the sum of squares of differences between computed and observed discharges. Of ten optimisation runs with different seeds, eight were able to locate the global minimum. The value obtained from the other two runs was only marginally inferior to the global minimum.
Similar work has been reported by Franchini (1996), who used a GA in combination with sequential quadratic programming to calibrate a conceptual rainfall-runoff model.

- Genetic algorithms have also successfully been used in groundwater problem.

Ritzel et al. (1994) solved a multiobjective groundwater pollution containment problem using a GA. In this paper, a simple GA and two variations of multiple objective GAs, a vector-evaluated GA (VEGA) and a pareto GA, were presented. The problem involved finding the set of optimal solutions on the trade-off curve between the reliability and cost of a hydraulic contaminant system.

McKinney and Lin (1994) reported on the combination of groundwater simulation models and a genetic algorithm to solve groundwater management problems: maximum pumping from an aquifer; minimum cost water supply development; and minimum cost aquifer remediation. The results showed that genetic algorithms can be used effectively and efficiently to obtain globally optimal solutions to these problems. The speedup of using the GA method on parallel computers with different number of processors was also discussed. It was concluded that for complex groundwater management problems a parallel implementation of the GA and the simulation model was likely be required.

- The GA technique has so far had few applications in reservoir systems optimisation.

Esat and Hall (1994) compared a GA with other optimisation methods, such as Discrete Differential Dynamic Programming (DDDP) for a “four-reservoir problem” that has become something of a benchmark for water resources system optimisation algorithms. The objective was to maximise the benefits from power generation and irrigation water supply subject to constraints on storages and releases from the reservoirs. The paper demonstrated the significant potential of the GA in water resources systems optimisation, and the advantages of the GA over standard dynamic programming technique in terms of computational requirements. It was concluded that DDDP is more effective than the GA
approach when applied to small problems. However, once the number of
decision variables exceeds four, the GA approach showed considerable economy
in both memory requirements and execution time.

Fahmy et al. (1994) also applied a GA to a reservoir system, and compared the
GA performance with that of dynamic programming. They concluded that GAs
had potential in application to large river basin systems.

Connarty and Dandy (1994) successfully applied genetic algorithms to the
massive combinatorial problem involving scheduling of reservoir capacity
expansion options.

Oliveira and Loucks (1997) used a GA to estimate effective operating policies for
multipurpose multireservoir systems. The research suggested that genetic
algorithms offer a practical and robust way of identifying effective operating
policies. Significant benefits were perceived to lie in the freedom afforded by
GAs in the definition of operating polices and their evaluation.

Wardlaw and Sharif (1999) evaluated several alternative GA formulations using a
modified nonlinear four-reservoir problem and a more complex ten-reservoir
problem. Particularly they studied ways to derive predefined operating policies
that indicate the total release from the system of reservoirs and the individual
reservoir storage targets as functions of time of the years and the existing total
storage volume in the system. The result achieved indicated that the approach
overcomes some of the limitations of many techniques based on more traditional
mathematical programming (constrained optimisation) models. It was concluded
that the most promising genetic algorithm approach for the four-reservoir
problem comprises real-value coding, tournament selection, uniform crossover
and modified uniform mutation. This work significantly extended the work of
Esat and Hall (1994) and led to consideration of the potential of the GA
application in real-time reservoir operation with stochastic inflow forecasts,
where the objective function is complex and the other techniques are difficult to
apply.
Sharif and Wardlaw (2000) applied a GA to the optimisation of multireservoir systems. A comparison of the genetic algorithm results with those produced by discrete differential dynamic programming was also presented. In all the cases considered, the genetic algorithm results are very close to the optimum and the technique appeared to be robust. All these studies compared their solutions with traditional, linear, nonlinear and/or mixed programming methods. Results show that these new global search methods can be more efficient and robust than traditional programming methods, especially for highly complex, nonlinear problems with local optima (Wang and Zheng 1998).

In brief, the performance of GAs has been documented on a wide range of difficult-to-solve problems from many different fields. The main attraction of genetic algorithms is their domain independence and robustness; that is, they are able to deal with a diversity of problems using essentially the same computer code with simple modifications. GAs can work with measures of system performance that may be nonlinear, nondifferentiable, and even discontinuous. The algorithms only require that any proposed solution can be evaluated based on some measures of performance, and that its performance can be quantified and compared to the performance of other solutions. In addition, GAs usually do not get trapped at sub-optimal local solutions, and can find a near-optimal solution with a high probability. Finally the parallel nature of genetic algorithms makes them appropriate for parallel processing (Oliveira and Loucks 1997). This may be especially attractive if evaluating the performance of any single solution takes considerable computer time.

GAs also have some drawbacks. The first shortcoming is that they converge slowly. Another one is that the GAs involve a large number of objective function evaluations. As GAs have traditionally been run on serial machines only, this function evaluation can be very time-consuming and it is often limited to searching near-optimal solutions (Morshed and Kaluarachchi 2000).
2.3.2 Shuffled Complex Evolution Method

The shuffled complex evolution method is a general-purpose global search strategy developed at the University of Arizona. It was designed to handle the various response surface problems encountered in the calibration of nonlinear simulation models, particularly the multilevel or nested optimisation problems encountered with conceptual hydrologic models (Duan et al. 1992). This search method conducts multiple concurrent searches within the parameter space. Each search is based on a complex of parameters, initially sampling from the search space randomly. At each iteration a simplex is randomly selected from each complex and allowed to evolve in a downhill direction using variant of a Nelder and Mead’s (1965) simplex method. Periodic shuffling is introduced so that new complexes are formed and the information gained by previous complexes is shared. Detailed explanations of the method can be found in Duan et al (1992, 1993, and 1994). In the past few years, many researchers have tested this method on a variety of models with overwhelming positive results. The algorithm is typically able to find the global optimum of the problem consistently and efficiently, whereas other optimisation methods either fail or provide inconsistent results (Gupta 1999).

Duan et al (1992) presented results that clearly established the nature of the problem of multiple optima in conceptual rainfall-runoff (CRR) models and showed that the optimisation problem is more difficult than had been previously thought. In addition to the previously known problems of parameter interaction, nonconvexity of the response surface, and discontinuous derivatives, the problem of multiple optima occurs on at least the following two scales. At the large scale, there is more than one broad region of attraction into which a search strategy may converge, while at the small scale, each major region of attraction contains a large number of minor local minima. The existence of widely distributed minor local optima on the response surface virtually guarantees that local search optimisation procedures will terminate prematurely with low probability of finding the optimal parameter set. In this paper, four global search procedures were evaluated on a research CRR model: the adaptive random search (ARS) method, the multistart simplex method (MSX), a combined
ARS/simplex method, and the shuffled complex evolution (SCE) method developed by the authors. The results indicated that both the MSX and the SCE methods were effective in consistently finding the global optimal solutions and the SCE was three times more efficient. The ARS and ARS/simplex methods were both found to be inadequate.

Sorooshian et al. (1993) applied the SCE and MSX global optimisation methods to the calibration of the Sacramento soil moisture accounting model of the National Weather Service River Forecast System. In the synthetic data study, a 100% success rate was obtained using the SCE algorithm, while none of the MSX runs were able to exactly locate the global optimum even with more than twice the number of function evaluations. In a real data study, the SCE method obtained consistently lower function values and more closely grouped parameter estimates, while using one-third fewer function evaluations than the MSX algorithm. The results indicated that the SCE method is a relatively consistent and effective method capable of locating the global optimum in the parameter space.

Duan et al (1993) compared the SCE method with a controlled random search method and a multi-start algorithm on seven well-established test functions and demonstrated the capability of the SCE. From their study on the test functions, it was observed that the larger the values of the number of complexes, the higher the probability of converging to the global optimum, but at the expense of larger number of model simulation.

Luce and Cundy (1994) compared two algorithms, the Simplex and SCE method to estimate parameters for a physically based infiltration and overland flow model. It was demonstrated that the SCE method was able to estimate physically reasonable parameter values reliably. Tanakamaru and Burges (1996) applied two local-type and six global-type optimisation methods to parameter estimation of a conceptual rainfall-runoff model with 16 unknown parameters. The eight methods were: Nelder and Mead’s simplex method, Powell’s conjugate direction method, Genetic algorithms (GA), GA with finer tuning using the simplex method, GA with Powell’s method, Shuffled complex evolution method, Multi-start simplex method, and Multi-
start Powell method. The consistency and efficiency of the parameter estimation was evaluated by numerical experiments using error-free synthetic data. The results showed that the SCE method provided the best performance after allowing for overall efficiency. The SCE method was then applied to a real problem using historical data of the Egienji Dam Basin in Japan with good outcomes.

Kuczera (1997) compared the SCE method with other probabilistic search methods, such as traditional genetic algorithm, multiple-start local search method and simplex and quasi-Newton algorithms. The case study involved the five-parameter SFB rainfall-runoff model. The robustness and efficiency of the SCE algorithm, demonstrated by Duan et al. (1992) and Sorooshian et al. (1993) for the Sacramento model, was corroborated in this case study for the SFB model. However, the GA failed to find the global optimum with high probability, most likely because a simple non-customised GA was used. The author suggested that GA performance could be enhanced by improving its algorithmic structure.

Gupta et al (1999) applied the SCE method to the calibration of Sacramento soil moisture accounting (SAC-SMA) streamflow forecasting model of the U. S. National Weather Service. The capability of the SCE automatic procedure was compared with the interactive multistage semi-automated method developed by Brazil (1998). The result indicated that the state of the art of automatic calibration methods had evolved to the point where such methods can be considered seriously as a viable alternative to the manual approach.

Despite the SCE method having a high probability of finding the global optimum, the method was found to occasionally terminate prematurely. Liong et al. (2001) improved the SCE method by systematically generating the initial population, instead of randomly generating it. The chances of the search getting trapped in local optima are reduced as indicated by the results on the seven test functions used. The convergence rate showed slight improvement.

Efstratiadis and Koutsoyiannis (2001) compared three optimisation methods by means of both mathematical applications and real-world problems. The first two
were the most popular applications related with hydrology and water resources, i.e. a simple genetic algorithm adapted from Goldberg (1989) and the shuffled complex evolution algorithm from Duan et al. (1992). The third one was a new simplex-annealing scheme, which incorporated the principles of simulated annealing in the well-known downhill simplex method. This scheme was very simple to implement and extended analysis proved that it was very effective in locating the global optimum as well as very efficient in terms of convergence speed. In most cases, the simple GA was neither effective nor efficient enough due to the simple formulation used. The shuffled complex evolution method proved to be robust and efficient and was recommended by the authors, especially when fast location of a good solution was desired.

In brief, the SCE method promises to be an effective and efficient optimisation method in consistently finding the global optimal solution. It is able to cope very well with rough, insensitive, and highly nonconvex objective function surfaces and is relatively unaffected by the presence of small local optima. It can be used to solve a broad class of complex problems, which the traditional optimisation methods usually have difficulty with. Similar to methods such as GA, the SCE method is flexible and easy to implement. During the past years, the SCE method has received more and more attention due to its ability of finding optimal solutions for hard-to-solve problems.

2.3.3 Simulated Annealing (SA)

The term ‘annealing’ comes from metallurgy (Metropolis et al. 1953). The central idea behind simulated annealing is an analogy with thermodynamics, specifically with the way liquids freeze and crystallize, or metals cool and anneal. At high temperatures, the molecules can move freely. As the temperature is slowly lowered, the molecules line up forming crystals, which represent the minimum energy state for the system. The objective function is analogous to the free energy of a thermodynamic system. A control parameter, analogous to temperature, is used to control the iterative optimisation algorithm until a state with a low objective function is reached. SA is a commonly used method for finding the global optimal solution.
The method has been applied to a number of optimisation problems that involve finding the optimum solution of a function of a very large number of independent variables (e.g. Kirkpatrick et al. 1983, Geman and Geman 1984, Aarts and Korst 1989).

Dougherty and Marryott (1991) used simulated annealing to optimise combinational groundwater management problems. The method was demonstrated by example applications to idealised problems of groundwater flow and selection of remediation strategy, including optimisation with multiple groundwater control technologies. They demonstrated the flexibility of the method and indicated its potential for solving groundwater management problems and difficult problems in other areas of hydrology. However, its development is relatively immature, so further performance improvements can be expected.

Sen et al (1992) discussed and compared three different algorithms for generating stochastic permeability fields. The algorithms considered two versions of simulated annealing, the Metropolis and heat bath algorithms, and the genetic algorithm. They showed that all algorithms performed equally well. For small problems, the Metropolis algorithm was clearly the fastest. For larger problems, the heat bath algorithm worked as well or better. The genetic algorithm converged very close to the global optimum, but usually required several trial runs to find an optimal population size. Its performance was highly dependent on the choice of population size, and probability of crossover and mutation. The main advantage of the GA method, however, is its amenability to parallel processing.

Ingber and Rosen (1992) compared genetic algorithms with Very Fast Simulated Re-annealing (VFSR) on six standard test functions involving relatively easy to difficult function optimisation. The test suite was designed to evaluate algorithms on functions that are continuous or discontinuous, convex or concave, unimodal or multimodal, linear or nonlinear, low or high dimensional, deterministic or stochastic. The preliminary results showed that VFSR is at least an order of magnitude superior to Genetic Algorithms in convergence speed and is more likely to find the global
optima during a time limited search. However, practical problems with large
dimensions remain to be investigated.

Marryott et al (1993) used simulated annealing to analyse alternative design
strategies for groundwater remediation at a contaminated field site. The simulated
annealing algorithm was combined with a field-scale, two-dimensional flow and
transport simulation model to analyse alternative remediation strategies. A series of
demonstration problems were presented using two different optimisation
formulations. The results of these experiments are generally consistent with the
results of the remedial pumping design implemented in the field. It was suggested
that the method can be applied to realistic groundwater management problems.
Unfortunately, the computational expense of simulated annealing is large, yet
comparable to other optimisation techniques.

Christakos and Killam (1993) used SA search algorithms for classifying spatially
distributed contaminant levels. It was found that the use of a SA algorithm could find
a good set of future sampling locations effectively. However, due to excessive CPU
time, a rather limited amount of experimentation with the annealing parameters was
attempted. It was recommended that computational efficiency must be improved to
allow more experimentation on the SA parameters.

Cieniawski et al (1995) compared the GA with simulated annealing (SA) in a case
study dealing with the multiobjective optimal location of a network of groundwater
monitoring wells under conditions of uncertainty. It was found that the GA had a
distinct advantage over the SA in that it was able to generate both convex and
nonconvex points on the trade-off curve, accommodate nonlinearities in the two
functions and not be restricted by the peculiarities of a weighted objective function.
Simulated annealing relies on a weighted objective function which can find only a
single point along the trade-off curve for each iteration, while the GA was able to
find a larger number of convex and nonconvex points on the trade-off curve in a
single iteration. The authors also suggested that the speedup of the GA can be
increased by parallelling the objective function evaluation.
Wang and Zheng (1999) developed groundwater management models using genetic algorithms (GA) and simulated annealing (SA). The objective functions were of a very general nature, incorporating multiple cost terms such as the drilling cost, the installation cost, and the pumping cost. For the three examples examined in this work, the GA and the SA based models yielded nearly identical or better solutions to those found by traditional optimisation methods.

In summary, the SA technique has a simple and universal structure and can be very efficient. However, compared to the GA and the SCE methods, the SA method has not seen the same exposure to complex problems in water resources systems analysis.

2.4 SUMMARY

The literature describing optimisation approaches in general and applications to reservoir operations in particular is extensive. Therefore, instead of detailed review of each individual approach, a short review was undertaken of traditional optimisation methods with emphasis on identifying the strengths and weaknesses of each approach in application to complex nonlinear stochastic systems. With regard to modern optimisation methods, emphasis has been given to some of the more commonly used methods, such as genetic algorithms, the shuffled complex evolution algorithm and simulated annealing. These modern optimisation methods have the potential to effectively deal with the nonlinear complex stochastic problems to be considered in this study. Overall, these methods overcome many of the major shortcomings of classical optimisation methods.

The following conclusions can be drawn:

1. The traditional optimisation approaches using linear, nonlinear, and dynamic programming have only been successfully applied to systems with known future inputs and/or low levels of complexity.

2. None of the traditional optimisation approaches found in the literature are able to deal comprehensively with the following aspects of reservoir planning and
operation problems - nonlinearity, stochastic inputs, and complexity (multiple reservoirs, multiple periods).

3. Modern stochastic optimisation approaches have the ability to locate solutions to combinational optimisation problems with greater efficiency than other techniques. They are also advantageous because they more easily accommodate the discontinuity, nonlinearities and complexities of real problems than traditional methods.

4. The GA and SCE methods have been proven to be efficient and effective in finding optimal solutions and have great potential for optimising large-scale complex problems. However, computational times for both methods will be the big issue especially when these optimisation methods are used in conjunction with Monte Carlo simulation.

5. There exist very few works dealing with optimisation of operation of urban water supply headworks systems using GA or SCE methods.
CHAPTER 3

GENETIC ALGORITHMS AND THE SHUFFLED COMPLEX EVOLUTION METHOD
3.1 INTRODUCTION

Modern probabilistic search methods have successfully solved large combinational problems. They can be coupled to a generalised simulation model to accommodate system complexity as well as stochastic inputs. They have been successfully used in a wide spectrum of problems. This chapter examines in some detail the features of such techniques, focusing on the two most widely used methods in hydrology and water resources, namely the genetic algorithm (GA) and the shuffled complex evolution (SCE) method.

3.2 GENETIC ALGORITHMS

Genetic algorithms are a class of search algorithms that simulate the process of natural selection inspired by Darwin's theory. They are also referred to as stochastic optimisation techniques. Stochastic optimisation designates a family of optimisation techniques in which the solution space is searched by generating candidate solutions with the help of a pseudo-random number generator. As the search proceeds, the probability distribution by which new candidate solutions are generated may change, based on results of trials found earlier in the search (Grefenstette 1986, Goldberg 1989, Wang 1991).

Many researchers have put forward different formulations and refinements to the traditional GA method. This diversity represents one of the difficulties facing potential users of genetic algorithms since it is hard to judge a priori which variations might be the best for particular applications. The variations use different data structures to describe the implementation of a single individual, different genetic operators to transform solutions, different methods to create an initial population or handling constraints, and different parameters such as population size, probabilities of selection, crossover, and mutation and number of generations (Montesinos et al. 1999). This section describes how a simple genetic algorithm operates and the way in which it can be adapted for use with a particular objective function.
3.2.1 A Brief History of Genetic Algorithms

The story of the development of the genetic algorithm can be traced back to the very beginning of the computer age. The early pioneers of computer science were as much interested in biology and psychology as in electronics (Mitchell 1997):

“Computers were applied not only to calculating missile trajectories and deciphering military codes but also to modelling the brain, mimicking human learning and simulating biological evolution. These biologically motivated computing activities have waxed and waned over the years. However, since the early 1980s, they have all undergone resurgence in the computation research community. The first has grown into the field of neural networks, the second into machine learning, and the third into so-called evolutionary computation, from which genetic algorithms (GAs) were invented.”

In the 1950s and 1960s, several computer scientists independently studied evolutionary systems with the motivation that evolution theory could be used as an optimisation tool for engineering problems. The idea behind it was to evolve a population of candidates solutions to a given problem, using operators inspired by natural genetic variation and natural selection. Based on this, the theory of evolutionary computing was introduced in the 1960s by Rechenberg (1973) in his work "Evolution Strategies". His idea was then refined by other researchers. The theory of Genetic Algorithms (GAs) was invented by John Holland, who with his students and colleagues refined it throughout the 1970s at the University of Michigan. This resulted in publication of "Adaptation in Natural and Artificial Systems" (Holland 1975).

Holland's original goal was not to design an algorithm to solve specific problems, but rather to study in detail the phenomenon of adaptation as it occurs in nature and to develop ways in which the mechanism of natural adaptation might be imported into computer systems (Holland 1975).

“It is a study of how systems can generate procedures enabling them to adjust efficiently to their environment. If adaptability is not to be arbitrarily
restricted at the outset, the adapting system must be able to generate any method or procedure capable of an effective definition.”

He also recognised the fundamental role of natural selection – survival of the fittest – in whatever programs and machines one might design. Inspired by the natural mechanism of selection and reproduction, Holland created an "intelligent" form of a random search that explores the solution space to find the more promising regions. His major innovation was to introduce a population-based algorithm with crossover, inversion, and mutation. A comprehensive text by Goldberg (1989) provides an extensive overview and explains recent research on the GA. Davis (1991) also reviewed many important applications of this class of algorithms. The algorithms rely on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions. They draw their power from the theoretical principle of implicit parallelism (Holland 1975). This principle enables highly fit individuals (or solutions) to propagate an increased number of offspring in successive generations and thus lead to a better solution.

In recent years, genetic algorithms have been developed into a powerful optimisation approach. Nonetheless, considerable research continues to be directed at GA applications. The basic concept of the GA is explained in the next section.

3.2.2 How GAs Work?

To apply a genetic algorithm to an engineering problem, one’s thinking must adjust to accommodate the GA paradigm. As stated by Goldberg (1989), the structure of genetic algorithms differs from the traditional search techniques in the following ways:

1. GAs typically work with a coding of the decision variable set, not with the decision variables themselves.

2. GAs search in parallel, using a population of decision variable sets, not a single decision variable set.
3. GAs are free from a particular model structure (Goldberg and Kuo 1987), and thereby require only an estimate of the objective function value for each decision variable set in order to proceed.

4. GAs use probabilistic search rules, not deterministic rules. They are randomised algorithms, in that they use operators whose results are governed by probability. The results for such operations are based on the value of a random number.

In keeping with genetics terminology, the decision space is referred to as the environment, the potential solutions to the optimisation problem are called chromosomes (or strings that represent a set of decision variables), the total number of solutions is called the population size. The iterations of the optimisation process are called generations. Simple genetic operations of reproduction, crossover, and mutation on populations are the essence of GAs. The GA proceeds by evaluating the best sets of chromosomes in the population at each generation. Therefore, the first step in a GA is to establish the initial set of solutions called the initial population. This may be done by either randomly generating a population or using some other initialisation in the search space. The objective function values (fitness) are then calculated for each population number and ranked. A second generation of potential solutions evolves by selecting individuals from the previous generation according to their objective function values. Individuals with high objective function values will not automatically survive; rather they will have a higher probability of being selected in the next generation. Following the selection, individuals mate and the genetic operators of crossover and mutation are applied stochastically to produce offspring. The new generation is thus an adaptation from the previous generation. When the second generation is created each individual's fitness value is evaluated and the entire process is reiterated. Through a number of generation cycles, the GA guides the population in the search space toward solutions of improved performance, yielding solutions more and more concentrated in the vicinity of the optima. The GA terminates if some stopping criteria (for example number of generations or improvement of the best solution) is satisfied. The following general description encompasses most of the important features of a standard genetic algorithm using the
three operators of reproduction, crossover, and mutation. Figure 3.1 shows a

Figure 3. 1 Flow chart describing a simple genetic algorithm

flowchart representation of a simple genetic algorithm.

1. Generate a random initial population and evaluate the objective function value of each decision variable in the population.

2. Create a new population by repeating the following steps until the new population is complete:

   • Selection - select two parent chromosomes (or solutions) from the population according to their fitness value. The fitter the fitness value, the greater the chance of being selected.
   
   • Crossover - with a predefined probability crossover the two parents to form two new offspring (children). If no crossover is performed, offspring are an exact copy of parents.
   
   • Mutation - with a predefined probability mutate new offspring.
   
   • Place new offspring in a new population.
3. If the stopping criteria are met, stop and return the best solution in the current population; otherwise go to step 2.

### 3.2.3 Encoding

The generic GA provides a framework within which there is considerable scope for variation. The first question to explore is how to represent solutions (or chromosomes). In other words, how should solutions be encoded? The genetic algorithms require encoding schemes to transform the vectors of decision variables to a structure that permits genetic operations. The most commonly used encoding schemes are binary strings, that is, strings of binary 0 and 1 bits (Goldberg 1989). Two solutions or chromosomes are illustrated in Figure 3.2.

<table>
<thead>
<tr>
<th>String 1</th>
<th>1101100100110110</th>
</tr>
</thead>
<tbody>
<tr>
<td>String 2</td>
<td>1101111000011110</td>
</tr>
</tbody>
</table>

**Figure 3.2 Illustration of binary string**

Each chromosome has one binary string. Each bit or group of bits within the string can represent some characteristic of the solution. Alternatively, the whole string represents one decision variable. In the latter case, a k-bit binary variable represents one variable $X_i$. The integer of the decoded binary variable ranges from 0 to $2^k - 1$ and can be mapped linearly to the variable range $[A_i, B_i]$. Other coding schemes, such as integer and real number coding, can also be used. Theoretical work done by Goldberg (1989) suggested that the performance of a GA is optimal when a binary coding is implemented. Binary encoding offers the maximum number of schemata, building blocks or binary templates that provide a means of comparing combinations of bits to achieve improved performance (Goldberg 1989, Wang 1991). A study done by Dandy and Connarty (1995) found that integer coding is more efficient and leads to equally good or better solutions.

Although argument exists about the choice of encoding, a traditional binary encoding scheme is used in this study. Each decision variable is encoded into a binary string of fixed length $k$. The entire population of such designs constitutes a
CHAPTER 3

A design (or chromosome or individual) refers to a set of decision variables that fully specify the optimisation problem.

3.2.4 Selection Methods

Next question regarding the GA implementation is how to select parents for crossover. This can be done in many ways using reproduction. The main idea is to select the better or fitter parents for crossover (in the hope that better parents will produce better offspring).

According to Darwin’s theory of evolution the best or fittest individuals should survive and create new offspring. Reproduction is an operator where an old string is copied into the new population based on that string’s objective function value or fitness. Under reproduction, fitter chromosomes are more likely to produce offspring. The problem is how to select these chromosomes from the parent population. There are many different ways to implement reproduction. For example, roulette wheel selection, tournament selection, and rank selection. Some of these are briefly described below:

Roulette Wheel Selection: This method is conceptually equivalent to assigning each individual a slice of circular roulette wheel which is proportional to the individual’s fitness. The roulette wheel is spun, the ball rests on a one wedge-shaped slice, where the corresponding individual is selected. This can be simulated by using following algorithm:

1. Sum the fitness value of all chromosomes in the population and yield a value $S$.

$$ S = \sum_{i=1}^{n} F(i) \quad (3.1) $$

where $F(i)$ is the fitness value for the $i^{th}$ individual, $n$ is the number of solutions in a population.

2. Calculate each chromosome’s percentage of population total fitness. A roulette wheel is then created where each current chromosome in the population has a roulette wheel slot sized in proportion to its fitness.
3. Choose a random number $r$ between 0 and $S$. Rank the fitness values and sum the ranked fitness values until the sum is greater than $r$. The individual of whose fitness value makes the sum greater than $r$ is the one to be selected.

The roulette wheel selection depends very much on the variance of fitness in the population. For example, if the best chromosome fitness is 90% of the entire roulette wheel then the other strings will have very few chances to be selected. This prevents the GA from exploring other regions of the search space, which can possibly cause a premature convergence (Goldberg 1989).

**Rank Selection:** In rank selection, the fitness rank of each individual is used instead of its absolute value of fitness. After each generation, the individuals in the population are ranked in increasing order of fitness values. The worst will have the fitness rank 1, the second worst has the rank 2, while the best will have the fitness rank $N$ ($N$ is number of individuals in the population).

The rank selection method helps to prevent the premature convergence. However, the possible disadvantage of this method is that the GA may be very slow to find highly fit individuals (Mitchell 1997).

**Tournament Selection:** Tournament selection is inspired by the competition in nature among individuals for the right to mate. Two individuals are randomly selected from the current population. The selected individuals are then pitted against each other in a tournament - the fitter one wins. The two winning individuals survive as parents and are subjected to the crossover and mutation to produce offspring. This process is repeated until the population size for the new generation is as large as the one from the previous generation.

Tournament selection is able to eliminate the random noise from the selection process and improves the efficiency of the GA search (Goldberg and Deb 1991, Kuczera 1997). The method has been more widely used than the roulette wheel and rank selection methods. In fact, tournament can be selected for more than two individuals and that tournament size is another parameter that can be selected in the

$$P(i) = \frac{F(i)}{S} \quad (3.2)$$
optimisation. Therefore, in this study, the tournament selection method is adopted with a minor modification, namely three individuals are randomly selected from the population in the hope of finding fitter individuals.

3.2.5 GA Operators

The crossover and mutation are the most important part of the genetic algorithm. The performance of the GA is influenced mainly by these two operators. Crossover and mutation are described in following subsections. A lesser-known operator, inversion, is then introduced.

3.2.5.1 Crossover

Crossover is the partial exchange of bits between two parent strings to form two offspring strings. It serves two complementary search functions (Goldberg 1989):

1. Exploitation which provides new decisions for further testing within the neighbourhood of the best solution in the current population;
2. Exploration which introduces decisions from the whole search space into the population to maintain global diversity in order to avoid being trapped at local optima.

Goldberg (1989) described in detail many different methods of performing the GA crossover operation, in which one point crossover is the most straightforward and commonly used method. Figure 3.3a illustrates how the one point crossover operation works. The crossover occurs if \( r < P_{\text{cross}} \), where \( r \) is a random number uniformly distributed between 0 and 1, and \( P_{\text{cross}} \) is a predefined probability for crossover. Next, a crossover point \( L \) is randomly selected, \( 1 < L < k-1 \), where \( k \) is the length of the binary string, and \( L \) is a uniformly distributed random integer between 1 and \( k-1 \). The crossover begins by selecting two individuals at random from the population. In the one point crossover the same location is selected on both parent strings. The two child strings are then created from the two parental strings by swapping the bits to the right of the cut.
Traditional one-point crossover of binary strings can be further modified (Kuczera 1997). Once the cut is made the offspring strings are swapped between the parents using either a direct (as shown in Figure 3.3a) or an indirect swap (as shown in Figure 3.3b). Therefore, another predefined probability $P_{\text{swap}}$ is needed to decide if the indirect swap is used. The indirect swap occurs if $r < P_{\text{swap}}$, otherwise a direct swap is used. Figure 3.3b illustrates the indirect diagonal swap operation. The indirect swap introduces more mixing than the traditional direct swap.

![Table of crossover results](image)

**(a)**

Parent string 1 | 1000 | 0101001 | Offspring string 1 | 1000 | 1101001
Parent string 2 | 1111 | 1101001 | Offspring string 2 | 1111 | 0101001

**(b)**

Parent string 1 | 1000 | 0101001 | Offspring string 1 | 1101001 | 1000
Parent string 2 | 1111 | 1101001 | Offspring string 2 | 0101001 | 1111

**Figure 3.3 Illustration of genetic crossover (a) direct and (b) indirect**

### 3.2.5.2 Mutation

Mutations are simply an insurance policy against the irreversible loss of genetic material (Goldberg 1989). Mutations are introduced into the population on a bit-by-bit basis at every generation to prevent the search from locking onto a local optimum.

Typically there is a low probability of mutation. The GA considers each string bit by bit in the new generation formed as a result of reproduction and crossover. For each bit of a binary string in the population, a random number, $r$, uniformly distributed between 0 and 1 is generated. If $r < P_{\text{mutate}}$, that bit is mutated by changing its value from either 0 to 1 or 1 to 0, as appropriate. Mutations help to prevent all solutions in a population from converging to a local optimum.
prematurely. Mutation changes randomly the new offspring. For binary encoding, Figure 3.4 illustrates the mutation operator.

| Original offspring 1 | 1101110011 | Mutated offspring 1 | 11001110011 |
| Original offspring 2 | 1011010110 | Mutated offspring 2 | 1011101100 |

**Figure 3.4 Illustration of mutation**

### 3.2.5.3 Inversion

An inversion operator is applied independently after the GA operators of tournament, crossover and mutation have produced the child population. The idea behind inversion is to produce orderings in which beneficial genetic material is more likely to survive. It provides a type of non-destructive noise that helps crossover to escape local maxima (Whitley 1987). For each string, if a random number is greater than the user defined probability $P_{\text{inversion}}$, the string remains unchanged, otherwise it is modified by the inversion operator. Figure 3.5 illustrates how this operator works. Under the inversion, two different bits are chosen at random along the length of the string and swapped.

| Original offspring | 1 0 0 1 1 0 1 1 | After inversion | 1 0 1 1 0 0 1 1 |

**Figure 3.5 Illustration of inversion**

### 3.2.6 Elitism

Creating a new population entirely based on new offspring can cause a loss of the best solutions' qualities from the previous population. The fittest individual of the population may not be selected for reproduction or may be crossed with unfit members and disappear from the population. An individual with those traits may not emerge again for hundreds of generations. To combat this problem, elitism is often used. The idea of elitism was first introduced by De Jong (1975). It forces the GA to
retain several of the best individuals at each generation. Several of the fittest members of each generation can be copied into the next generation untouched, ensuring that they will survive until they are replaced by even fitter members. This is known as an elitist strategy (De Jong 1975).

Exploitation and exploration are the two most important issues in the search process of the GA (Whitley 1989). Exploitation means that the decision variables from the already discovered promising search areas are exploited, while exploration refers to the promising new areas of the search space ready to be explored. Selective pressure is the degree to which the better individuals are favoured in the exploitation process. In contrast, exploration helps to maintain population diversity. The selective pressure drives the GA to improve population fitness over succeeding generations. The convergence rate of a GA is largely determined by the selective pressure and population diversity. In general, higher selective pressures result in higher convergence rates. However, if the selective pressure is too high, there is an increased chance of the GA prematurely converging to a local optimal solution because the population diversity of the search space to be explored is lost, and the vice versa.

As noted in section 3.2.5.1, the traditional GA theory completely places the burden of exploitation and exploration on crossover, forcing a tradeoff between population diversity and selective pressure. Increasing the selective pressure tends to reduce diversity but increases search speed, whereas decreasing the selective pressure helps maintain diversity but results in a slower, though more robust, search. According to Eshelman and Schaffer (1991), some of the burden of preserving good genetic material (schemata) can be transferred from the crossover to the selection procedure. One such method is an elitist strategy called the Population Selection Strategy (PSS) in which only those parents are replaced that are less fit than the newly created offspring.

In this study, a similar strategy is applied to preserve good genetic material and pass it on to the next generation. After the child population is created by the GA operators, the members of the child population are then compared with their parents.
Those members from the child population that are worse than their parents are replaced with the best members of the parent population. Therefore, the preservation of good genetic material from the parent population is independent of crossover, which enables the crossover to concentrate on maintaining population diversity. The crossover is no longer both the sources of exploitation and exploration, thereby enabling use of a high crossover probability to maintain population diversity.

The PSS strategy is adopted in this study. Because this is a form of elitism, De Jong’s concept of elitism will not be included in the GA formulation in this study.

3.2.7 Assignment of GA Parameters

The success of the GA depends on specification of several parameters: population size, number of generations, and the probabilities of crossover and mutation. It has been suggested that good GA performance requires the choice of high crossover and low mutation probabilities and a moderate population size (Goldberg 1989). However, it is difficult to predict how the various parameters interact with each other (Grefentette 1986). Indeed, there is no general theory for selecting optimal GA parameters.

*Population size:* The population size affects both the ultimate performance and the efficiency of GAs. It is a very sensitive parameter for the GA process (Halhal et al. 1997). GAs generally perform poorly with a very small population because the population provides an insufficient sample size and will converge rapidly to a local optimum. A large population, on the other hand, requires more evaluations per generation, possibly resulting in an unacceptably slow convergence. McKinney and Lin (1994) reported that populations of 50 to 100 individuals taken through 10 to 20 generations have a high probability of finding optimal or near optimal solutions. However, the best value will depend on the size of the total search space.

*Crossover rate:* The crossover rate controls the frequency with which the crossover operator is applied. The higher the crossover rate, the more quickly new structures are introduced into the population. If the crossover rate is too high, high performance structures are discarded faster than selection can produce
improvements. If the crossover rate is too low, the search may stagnate due to the lower exploration rate. This study uses the crossover in conjunction with the population selection strategy. It is recommended that a very high crossover rate be used with this strategy (Eshelman and Schaffer 1991). The effect of a high crossover rate in combination with the PSS on GA performance will be further examined in Chapter 4.

*Mutation rate*: Mutation plays a secondary role for increasing the diversity of the population. When using reproduction and crossover, some potential useful genetic material might be lost. The mutation operator protects against such irrecoverable premature loss. DeJong (1975) originally suggested that a mutation probability inversely proportional to the population size would be enough to prevent the search from locking into a local optimum. Recently, considerable attention has been focused on formulating GAs with adaptive and non-uniform mutation probabilities (Michalewicz 1992). A lower level of mutation prevents any given bit position from remaining forever converged to a single value in the entire population. A high level of mutation rate yields an essentially random search.

### 3.2.8 Formulation of a Customised GA

The GA represents a class of evolutionary search algorithms. There is considerable scope for customising the GA. This section describes the adaptations to the standard GA used in this study:

Randomly generate an initial population $p$ from the search space using a uniform sampling distribution, where $p = m \times n$, $m$ is the population members and $n$ the number of decision variables ranging between 0 and 1.

1. Calculate the objective function value of each population member.
2. Encode each decision variable into a binary string of the fixed length $k$.
3. A tournament selection is used to generate the new strings of the next generation. The GA terminates if all the new designs are the same as the designs from the previous population.
4. A modified one-point crossover is performed on each pair of new strings with predefined probability $P_{\text{cross}}$ and $P_{\text{swap}}$. Mutations are carried out with given probability $P_{\text{mutate}}$ on a bit-by-bit basis in the strings. An inversion is then performed based on a pre-defined probability $P_{\text{inversion}}$ for each string.

5. Decode the strings into their real value equivalents.

6. Evaluate the objective function of each population member.

7. Apply Population Selection Strategy to determine the population members for the next generation.

8. Steps 2 to 8 are repeated until stopping criteria are satisfied. The stopping criteria are defined as follows: the GA stops if the best design does not improve over a predefined number of generations (5) or if the number of evaluations exceeds a predefined maximum (10000).

### 3.3 THE SHUFFLED COMPLEX EVOLUTION METHOD

During the last three decades, a substantial body of research has documented the problems encountered during model calibration, especially with conceptual rainfall-runoff (CRR) models. However, only limited success has been achieved in overcoming these problems (Sorooshian et al. 1993). According to Duan et al. (1993), there are five major characteristics complicating the optimisation problem, which are: (1) several regions of attraction, (2) numerous local optima, (3) roughness of the objective function surface, (4) poor and varying sensitivity of parameters, and (5) nonconvex response surface. The primary conclusion of the study from Duan et al. (1992) was that existing optimisation techniques were not powerful enough to deal with the objective surface encountered in the model calibration. To overcome the above problem, Duan et al. (1992) developed the shuffled complex evolution (SCE) method for global optimisation, which promises to be effective and efficient for a broad class of problems. This method incorporates the best features from several existing methods that have proved successful for global optimisation, including the concept of competitive evolution, the combination of random and deterministic strategies, the concepts of controlled random search, and complex shuffling. A lucid
explanation of the theory underlying the SCE method is presented by Duan et al (1992, 1993). A general description is given in this section.

In essence, SCE begins with an initial population of points sampled randomly from the feasible space. The population is partitioned into one or more complexes, each containing a fixed number of points. Each complex is allowed to evolve based on a competitive evolution technique that uses the simplex method to direct the search in the correct direction. Periodically, the entire population is shuffled and points are reassigned to new complexes to enable information sharing. As the search progresses, the entire population tends to converge toward the neighbourhood of the global optimum, provided that the initial population size is sufficiently large.

The SCE strategy is described below:

1. Generate \( s = p \times m \) designs randomly from the search space using a uniform sampling distribution, where \( p \) is the number of complexes each having \( m = 2n+1 \) designs with \( n \) being the number of decision variables.
2. Calculate the objective function value of each design.
3. Sort all designs in ascending order of their corresponding objective function values and store them in an array \( D \).
4. Sequentially partition \( D \) into \( p \) complexes \( A_1, A_2 \ldots A_p \) such that \( D_1 \) is assigned to complex 1, \( D_2 \) to complex 2 and so on. Each complex contains \( m \) designs. This ensures that the whole population can be shuffled and the complexes reformed in order to share the information obtained independently by each complex.
5. Evolve each complex \( A_k \), \( k = 1 \ldots p \), independently using the competitive complex evolution (CCE) algorithm. The CCE algorithm is based on the Nelder–Mead simplex algorithm (1965) and described in detail by Duan (1993). This produces a new set of design.
6. If the stopping criteria are not met, go to step 2. The SCE search will stop if the number of evaluations exceeds a predefined maximum (10000), or if the smallest objective function value does not improve over 2 consecutive generations \( |f_i(j) - f_i(j-1)| < \epsilon \).
\[ f_i(j-1) / f_i(j-1) < 10^6, \] where \( f_i(j) \) is smallest the objective function value for \( j^{th} \) generation.

The key component of the SCE method is the CCE algorithm. This algorithm is based on the Nelder and Mead Simplex (1965) downhill search scheme and was described in detail by Duan et al (1993). Figure 3.6 illustrates this algorithm graphically using a two-dimensional example. The contour lines represent a function surface with a global optimum located at [4,2] and a local optimum located at [1,2]. The dots indicate the location of the sampled points in a complex before the evolution step is taken. A subcomplex containing \( q \) points (\( q \) equals 3 and forms a triangle) is selected based on a trapezoidal probability distribution to initiate the evolution step. The symbol * represents the new points generated by the evolution steps. Figures 3.6 (a), (b), and (d) illustrate the reflection step, which is implemented by reflecting the worst point in a subcomplex through the centroid of the simplex. The worst point is discarded and replaced by the new reflected point if the reflected point has a lower objective value than the worst point. In Figure 3.6 (c), the new generated point does not improve the objective function value, the reflection step is then rejected and a so-called ‘contraction step’ is used to get a new point. The new point is half way between the worst point and the new reflected point. Figure 3.6 (e) shows a “mutation step’ to replace the worst point of the subcomplex by randomly choosing a point in the feasible parameter space. It is done if the new reflected point is outside the feasible parameter space. Figure 3.6 (f) shows the final complex after several evolution steps, indicating the obvious improvement of the newly generated points in a complex.

The SCE method contains many probabilistic and deterministic components that are controlled by algorithmic parameters. Similar to the GA method, these parameters have to be chosen carefully. Duan et al. (1994) presented the results of several experimental studies using different algorithmic parameter setups. All recommended values from Duan et al (1994) for the algorithmic parameters are used as default values in this study, such as \( m = 2n+1, p = n+1 \). The most sensitive algorithmic parameter is the number of complexes \( p \). Duan et al’s experiments
showed that the greater the degree of difficulty of the problem, the larger the number of complexes required to find the global optimum.
Figure 3.6 Illustration of the evolution steps taken by each complex (from Duan et al. 1993)
The main strength of the SCE method comes from its periodic shuffling and reforming of the complexes which provides a global perspective of the objective function surface to each complex – this strategy enhances the algorithm’s chances of locating the global optimum. The SCE approach has been shown to be a robust and efficient method over a range of optimisation problems (Duan et al 1994).

3.4 SUMMARY

In this chapter, two probabilistic search methods, the genetic algorithm and the SCE method have been presented in detail. Although the GA and the SCE methods are both robust global probabilistic search methods that require no information about gradients and are capable of handling constraints, they are quite different in nature. The GA uses search strategies similar to the mechanisms of biological natural selection and evolution. It codes decision variables as binary strings and uses the three operators of reproduction, crossover, and mutation to search for the optimal decisions. In contrast, the SCE method uses a probabilistic adaptation of the Nelder-Mead simplex method along with shuffling of simplexes to maintain a global perspective on the search. Although SCE does employ some of the probabilistic evolutionary ideas embedded in GAs, these are of secondary importance.

Although both methods proved to be successful in solving many complex problems, their performance in optimising water supply headwoks system is largely unknown. The evaluation and comparison of these two optimisation methods is presented in the next chapter.
CHAPTER 4

ASSESSMENT OF OPTIMISATION METHODS USING A SIMPLE CASE STUDY EMPLOYING MONTE CARLO SIMULATION
4.1 INTRODUCTION

Water supply headworks systems are characterised by their complexity and stochastic forcing. Typically they have multiple users, possibly with different objectives and risk tolerances, and also multiple sources with different levels of quality. As discussed in Chapter 2, traditional optimisation approaches using linear, nonlinear, and dynamic programming have only been successfully applied to systems with known future inputs and/or low levels of complexity. They have been unable to adequately cope with either the complexity or the stochastic behaviour typical of large headworks systems.

It is widely accepted that simulation models in conjunction with the Monte Carlo approach provide the most realistic assessments of future system behaviour. Unfortunately the Monte Carlo approach does not easily interface with traditional mathematical programming methods. Modern probabilistic search methods have successfully solved large combinatorial problems and are well suited to interfacing with Monte Carlo simulation models. They can be coupled to a generalised simulation model to accommodate complexity as well as stochastic inputs. In Chapter 3, two modern probabilistic search methods, genetic algorithms and the shuffled complex evolution method, have been described in detail and formulated.

Although the GA and SCE methods have been shown robust and efficient for various optimisation problems (Duan 1994, Wang 1991), there have been few studies involving the optimisation of complex headworks systems subject to stochastic forcing. The relative performance of probabilistic search methods for optimising water supply headworks systems evaluated using Monte Carlo simulation is largely unknown. Perhaps the main deterrent to such studies is the massive computational burden. For example, to optimise the headworks system for Sydney, which consists of a network of nine major reservoirs and an inter-basin transfer scheme, a Monte Carlo simulation involving 1000 replicates over a 50-year planning horizon takes about 310 CPU seconds on a 750 MHz Pentium III processor. A probabilistic search method optimising 5 to 10 decisions may require evaluation of the order of a few thousand trial solutions to obtain a near-optimal solution. To perform 1000
evaluations would take about 3.6 days on a 750 MHz Pentium III processor! Clearly, this class of optimisation problem requires supercomputing resources, as issue explored in Chapter 5. However, before committing substantial supercomputing resources to such an optimisation problem, it is prudent to seek insight by studying a problem deemed sufficiently rich to identify the main challenges but simple to be computationally manageable. In this chapter, a simple case study is constructed to derive insight from such a problem about:

- The nature of the objective function to be interrogated by probabilistic search methods;

- The efficiency and robustness of probabilistic search methods, particularly the GA and SCE methods.

This chapter is organised into two sections. Firstly, it derives insight about objective functions typical of those to be encountered when optimising a system whose performance is to be evaluated using Monte Carlo simulation. It considers a simple case study involving the optimisation of three decisions associated with a single reservoir system using a risk-based economic performance criterion. Armed with important insights about the objective function the second section focuses on comparing the efficiency and robustness of the GA and SCE search methods.

4.2 URBAN WATER SUPPLY HEADWORKS SYSTEM CASE STUDY

The ability of an optimisation method to converge efficiently to a near-optimal solution depends on the nature of the objective function. For example, gradient-based search methods typically rely on the objective function surface being reasonably smooth and unimodal. In view of the massive computational effort required to realistically optimise even a moderate number of decisions in a headworks system it is important to have insight about the salient characteristics of the objective function as this will guide the selection of search methods.

To derive the necessary insight, a hypothetical urban water supply headworks system with three decision variables is considered. The system consists of one reservoir fed by an unregulated stream and supplying an urban demand zone for
which a high level of reliability is required. Typical of such systems, operating rules and objective functions exhibit threshold behaviour. The simplicity of the case study makes exhaustive testing computationally feasible. Nonetheless, the case study captures many of the key elements of headworks systems.

### 4.2.1 Decision Variables

Three decision variables are the object of the optimisation. Two of these variables are associated with the demand restriction policy and the third determines the size of the storage. Figure 4.1 illustrates the demand restriction rules typical of Australian practice. It tells at what storage level the restrictions are imposed and if so, how much demand is reduced. If the reservoir volume is in the top zone 1, then no restrictions are imposed, whereas if the volume falls into zone 2 restrictions are imposed. In this study, two reservoir storage zones are considered which requires one trigger storage level to be assigned. This trigger level is denoted by the fraction $Trig$. When restrictions are imposed, it is necessary to assign what fraction of demand is to be supplied - this fraction is denoted by $\alpha$.

Formally, the restricted supply $D_{rest}$ during time interval $t$ is expressed as follow:
where $D_i$ is the unrestricted demand, $\alpha$ is the supplied fraction of unrestricted demand, $0<\alpha<1$, $S_t$ is the reservoir volume at the start of time interval $t$, $C*S_{\text{max}}$ is the reservoir capacity with $C$ lying between 0 and 1 and $S_{\text{max}}$ being the maximum permitted design capacity, and $Trig$ is the fraction of storage capacity below which restrictions are imposed, $0<Trig<1$. The reservoir capacity fraction $C$, the restricted demand fraction $\alpha$, and the trigger level $Trig$ are the three decision variables to be optimised in this study.

### 4.2.2 Monte Carlo Simulation of An Urban Water Supply System

A Monte Carlo simulation is used to simulate the stochastic behaviour of the urban headworks system. For the $i^{th}$ replicate and the $t^{th}$ time interval, the following steps are executed:

1. Given the reservoir volume $S_i^t$ at the start of the time interval $t$, determine the restricted supply $D_{\text{rest}}$ using equation (4.1).

2. Let $q_i^t$ be a stochastically generated streamflow volume. The volume at the end of the time step $S_{i+1}^t$ is

   \[
   S_{i+1}^t = \begin{cases} 
   C*S_{\text{max}} & \text{if } q_i^t + S_i^t - D_{\text{rest}} > C*S_{\text{max}} \\
   0 & \text{if } q_i^t + S_i^t - D_{\text{rest}} \leq 0 \\
   q_i^t + S_i^t - D_{\text{rest}} & \text{otherwise} 
   \end{cases} \quad (4.2)
   \]

3. Given $S_{i+1}^t$, determine the actual supplied water $D_{\text{sup}}$ using

   \[
   D_{\text{sup}} (i,t) = \begin{cases} 
   D_{\text{rest}} & \text{if } S_{i+1}^t > 0 \\
   q_i^t + S_i^t & \text{if } S_{i+1}^t \leq 0 
   \end{cases} \quad (4.3)
   \]

4. Compute the drought penalty $C_D$ if the system fails to supply the required demand according to
This penalty function, though arbitrary, qualitatively reflects the economic/social consequences of demand shortfalls in Australian urban areas (Dandy 1992). Restrictions on the first 40% of demand are assumed to affect external domestic consumption which is largely in the form of garden watering and which arguably has a relatively small economic disbenefit. However, once restrictions exceed 40% of demand, it is assumed that commercial and industrial activity may be significantly impacted with loss of wages and production and that inhouse domestic consumption is restricted with major inconvenience to the household sector.

10000 streamflow replicates each 105 years long were used in the Monte Carlo simulation. The reservoir is assumed to be full at the start of the each replicate. To avoid the complication of seasonal variation an annual time step was used. Annual streamflow was sampled from a normal distribution with mean 150000 ML and standard deviation 30000 ML. Annual unrestricted demand $D_t$ was 120000 ML, whereas the maximum design capacity $S_{\text{max}}$ was 500000 ML.

### 4.2.3 Objective Function

The headworks optimisation problem seeks to minimise the objective function that contains two components. The first is the cost of constructing a reservoir with capacity $C \times S_{\text{max}}$. The second is the present worth of the expected penalty due to failure to meet demand during periods of drought. The objective function is:

$$OF = \frac{C \times S_{\text{max}}}{10000} + \sum_{t=1}^{n_{\text{year}}} p_w(t) \frac{1}{N} \sum_{i=1}^{N} C_D(i, t)$$  \hspace{1cm} (4.5)$$

where the subscript $t$ ranges from year 1 to year 105, $i$ represents the replicate number, $N$ is the number of replicates, and $p_w(t)$ is the present worth defined as $(1+r)^{-t}$ with $r$ being the discount rate taken as 5% in this study. The constant 10000 was selected to ensure that a meaningful trade-off between decision variables could be achieved in the case study.
4.2.4 Investigation of Objective Function Surface

The simplicity of the case study enabled exhaustive enumeration of the objective function surface by computing objective function over a finely spaced grid spanning the decision space. Figure 4.2 presents three subspaces of objective function evaluated using 1000, 5000 and 10000 replicates. In the Figure 4.2, the decision variable \( C \) is fixed at 0.05, which is close to its optimal value. The surface was evaluated on a regular grid with 0.01 spacing in the \( \alpha \) and \( Trig \) subspace with \( \alpha \) ranging between 0.6 to 1.0 and \( Trig \) between 0.01 to 1.0.

What is immediately evident is the irregular nature of the surface. The objective function surface over which the searches were conducted differs markedly from the smooth surfaces one typically encounters. It is characterised by many piecewise-flat regions separated by steep steps. Figure 4.2 also reveals that the number of the replicates used in the Monte Carlo simulation has a strong influence on the shape of the objective function. As the number of the replicates decreases, the objective function surface tends to have more extensive regions separated by sharper discontinuities. The presence of these piecewise-flat regions is attributed to:

1) The threshold nature of the operating rules; and

2) The fact that only a small proportion of the replicates experience severe drought stress.

In fact only about 20 out of 10000 replicates experienced water shortages exceeding 60% and only about 450 experienced shortages in the range of 40 to 60%. The high reliability of the system results in relatively few drought events that generate economic penalties with the consequence there is little opportunity to smooth the discontinuities introduced by the operating rule thresholds.

As indicated, the number of replicates can strongly influence the extent of the flat regions. The absolute number of replicates experiencing severe drought will decrease, as the number of replicates is decreased and thereby exaggerate the size of the flat regions. These considerations suggest that, for the same objective function value obtained from the search method, the decision variables can be very different.
1000 replicates

5000 replicates

10000 replicates

Figure 4.2 Objective functions surface for the decision subspace $\alpha$ and $\text{Trig}$ with $C = 0.05$ for different number of replicates.
depending on the spatial extent of the flat region. They also suggest that traditional gradient-based methods that rely on gradient information are likely to be in effective – once located on any extensive flat region, the search method is likely to terminate its search, most likely prematurely. Even robust derivative-free methods such the Nelder-Mead simplex method are likely to converge prematurely - for instance if the whole Nelder-Mead simplex was located on a single flat region, it is unlikely to progress any further. It appears that a key requirement for a robust search technique is that it have a mechanism to disregard local gradient information and have the ability to jitter on the flat regions in order to ‘fall off’ the edge of the flat region and slide onto another flat region.

4.3 EFFICIENCY AND ROBUSTNESS OF PROBABILISTIC SEARCH STRATEGIES

The piecewise flat behaviour of the objective function surface has the potential to make search methods converge prematurely, possibly at a markedly sub-optimal location in decision. The prospect of premature convergence could render the optimisation process a pointless exercise. Therefore, in assessing potential search methods it is essential to focus not only on efficiency but also on robustness (that is, the reliability of the search method to converge to near-optimal solutions).

The need for the search method to robustly negotiate extensive flat regions on the objective function surface narrows the choice down to methods which literally can “fall off” flat regions. Probabilistic search methods use randomised rules to guide the search and thus, in principle, possess the ability to negotiate flat regions. This section assesses two promising probabilistic search methods, the so-called genetic algorithm (GA) and the shuffled complex evolution (SCE) algorithm, which were discussed in Chapter 3.

4.3.1 Comparison Of The Two Search Methods

This section compares the performance of the GA and SCE methods for the case study described in Section 4.2, which involves three decisions: the reservoir capacity fraction $C$, the restricted demand fraction $\alpha$, and the trigger level $Trig$. Each method
was tested by running 20 trials with each trial using a different random number seed. To ensure consistency, the same streamflow series were used for evaluation of the objective function in all trials.

Two measures are used to assess performance. Efficiency is a measure of the computational effort required to find a near-optimal solution. It is defined by the number of evaluations required to convergence. Robustness is a measure of the ability of the search method to converge to a near-optimal solution. In this study, a trial is deemed a success if \((OF(i) - OF(best))/OF(best) < 0.01\), \(i = 1, \ldots, 20\) where \(OF(i)\) is the best objective function value by \(i^{th}\) trial, and \(OF(best)\) is the overall best objective function value. Robustness is defined as the percentage of successful trials.

For the SCE method, the only tuning parameter, the number of complexes \(p\), was set equal to the number of decision variables. The GA was run using a population size of 60 or more and a probability of an inversion of 0.9. The probability of a crossover was set at 0.9 with the probabilities of a diagonal swap and mutation being 0.3 and \(1/(\text{population size})\) respectively. The probability of the inversion was set at 0.9.

Figure 4.3 presents the lowest objective function value for 10000 replicates found by the GA and SCE methods for each of the 20 independent trials – the objective values are ranked in ascending order to assist interpretation of robustness. It was found that both methods locate near-optimal solutions in the same region of the search space. In the case of a population size of 80, the GA method consistently found a near-optimal solution for all 20 trials. The SCE and GA methods with population size of 60 have similar robustness, with 80% of trials converging to the near-optimal solutions. Over 20 trials the average number of GA evaluations required using populations of 60, 70 and 80 was 960, 1160 and 1280 respectively, whereas for the SCE the average was 710. The results suggest that the GA may be more robust than the SCE provided appropriate algorithmic parameters are selected, whereas the SCE is clearly more efficient than GA.
Figure 4.3 Best objective function values obtained by GA and SCE over 20 runs (The legend pXX-GA indicates the GA population size is XX).

4.3.2 Effect of Number of Replicate

Table 4-1 compares the performance of the GA and SCE over 20 trials for different number of replicates. The average objective function value and the average number of evaluations are reported only for the successful trials. Both search methods demonstrated similar success rates at locating near-optimal solutions. The GA yielded marginally better solutions for 4 out of 5 different number of replicates, while the SCE required fewer evaluations for all 5 runs. There is a trend for the average number of evaluations to increase with increasing the number of replicates. Figure 4.2 showed that as the number of replicates increases the objective function surface exhibits more but smaller flat regions. Presumably the search methods have a smaller chance of finding a near-optimal flat region and hence require more evaluations.
Table 4-1 Search method performance as a function of number of replicates

<table>
<thead>
<tr>
<th>Replicate length</th>
<th>GA(^1)</th>
<th>SCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average Evaluations</td>
<td>% success</td>
</tr>
<tr>
<td>1000</td>
<td>642</td>
<td>90</td>
</tr>
<tr>
<td>3000</td>
<td>714</td>
<td>85</td>
</tr>
<tr>
<td>5000</td>
<td>741</td>
<td>90</td>
</tr>
<tr>
<td>7000</td>
<td>904</td>
<td>90</td>
</tr>
<tr>
<td>10000</td>
<td>960</td>
<td>95</td>
</tr>
</tbody>
</table>

\(^1\)Population size = 60 and inversion rate = 0.9.

4.3.3 Effect of GA Parameters on Performance

The success of the GA depends on the choice of several parameter values. Goldberg (1989) suggested that good GA performance require the combination of the following: a high probability of crossover, a low probability of mutation and a moderate population size. Many different sets of GA parameter values were tested during the case study. The same probabilities used in Section 4.3.1 for a diagonal swap and mutation were used in the following trials.

Figure 4.4 shows the effect of crossover rate on the GA performance. The plot presents the lowest objective function values in ascending order for the 20 independent trials using 10000 replicates. Compared to the Figure 4.3 where a crossover rate of 0.9 was used, the GA’s robustness decreases markedly as the crossover rate decreases. This illustrates the importance of a high crossover rate to vigorously explore the decision space to avoid pre-mature convergence on a flat region.

Figure 4.4 also illustrates the importance of using the PSS. The line labelled noPSS represents a GA run with a crossover rate of 0.9, population of 80 but without
PSS. A major loss of robustness is evident. It is concluded that a very high crossover rate needs to be used to adequately explore the decision space and ensure robustness, but must be used in conjunction with PSS to preserve good solutions – this is consistent with the findings made by Eshelman and Schaffer (1991). Therefore, the crossover rate was set to 0.9 as the default value for the rest of trials.

![Figure 4.4 Effect of crossover and PSS (the legend pXXCrY indicates the population size XX and the crossover rate Y/10)](image)

Figures 4.5a to 4.5d show how, for a given population size, GA performance depends on the inversion rate. It can be seen from Figure 4.5a that the GA is robust when the population size is large (100) with the inversion rate having little influence on robustness. However, for a moderate population size of 60 (see Figure 4.5b), the robustness of the GA improves with increasing inversion rate. A relatively high inversion rate (≥0.7) is required to provide sufficient diversity in the sample population. Figures 4.5c and 4.5d show the GA performance for a population size of 50 and 40 respectively. For a population size of 50, the GA fails to converge to a near-optimal solution in 25% of the runs. For a population size of 40, the GA fails to converge in 40% of the trials. For these smaller populations, only the highest inversion rate appeared to have a beneficial effect on robustness. It appears that
selecting too small a population size compromises the robustness of the GA, but preferably closer to 60.
Figure 4.5 Best objective function values obtained by GA over 20 runs for different algorithmic parameter combinations (The legend pXXvY indicates the population size is XX and the inversion rate is Y/10).
On the other hand, it appears that too large a population size will decrease GA efficiency (Grefenstette 1986). Therefore, a moderate size of 60 with a high rate of inversion appears necessary to produce robust and efficient performance. From these considerations, the following GA parameters appear suitable for the case study and presumably for related problems: Crossover rate \( \geq 0.9 \); mutation rate = \( 1/(\text{population size}) \); inversion rate \( \geq 0.7 \); and population size in the range of 60 to 100.

4.4 DISCUSSION

Both the GA and SCE methods have comparable robustness for locating near-optimal designs for the simple urban water supply headworks system evaluated using Monte Carlo simulation. However, the SCE appears to be computationally more efficient than GA requiring up to 25% fewer evaluations. This is an encouraging result given the piecewise-flat nature of the objective function surface. Nonetheless, the efficiencies of both methods are not adequate to solve real problems in an acceptable period of time. The quest for efficiency remains extremely important for two reasons:

1. For high reliability systems, the number of replicates required to obtain accurate risk-based performance estimates is considerable. For example, to estimate with 95% confidence the probability of water shortages to within \( \pm 0.01 \) of the true value of 0.05 requires over 1800 replicates. If an accuracy of \( \pm 0.001 \) is required the number of replicates jumps to over 180000!

2. As system complexity increases the number of decision variables requiring optimisation will increase. This will increase the number of objective function evaluations required to find a near-optimal solution. Clearly, the issue of search efficiency is of critical importance. The challenge is to improve the efficiency of both the search methods and the Monte Carlo simulation to make such optimisation practicable.

The major deterrent to the use of these optimisation methods in water resource systems has almost certainly been associated with computing requirements. To appreciate the magnitude of the computational effort, a 1000-replicate simulation for
the 9-reservoir Sydney (Australia) water supply headworks system takes about 310 CPU seconds on a 750 MHz Pentium III processor. Typically several thousand of evaluations are required to obtain a near-optimal solution. Clearly, in such instances, the computational efficiency of the search method will be of critical importance. If realistic optimisation of operating rules is to be realised there appears to be little choice but to use Monte Carlo simulation. The challenge is to improve the efficiency of both the search methods and the Monte Carlo simulation to make such optimisation practicable.

One obvious improvement would be the employment of parallel computing. One of the most distinct features of GAs is their inherent parallelism (Goldberg 1989). In a parallel-computing environment, the objective function computation for the entire population can be performed concurrently because each individual is independent of the others. In contrast, the SCE approach partitions a population into $p$ complexes, where $p$ is typically equal to the number of decision variables. The opportunity to exploit parallelism within each complex is limited. This is because the search follows an algorithm similar to the Nelder-Mead Simplex method, which requires sequential objective function evaluation. It therefore appears that despite the somewhat superior efficiency of the SCE, the inherent parallelism of the GA will offer a huge advantage in practical parallel computing applications.

**4.5 SUMMARY**

Realistic optimisation of the operation and planning of urban water supply headworks systems requires that the issues of complexity and stochastic forcing be addressed. The only reliable way of accomplishing this is to use simulation models in conjunction with the Monte Carlo method which generates multiple hydro-climate replicates. To appreciate the challenge of this optimisation problem a case study involving a simple headworks system was presented. The system consisted of one reservoir and had three decision variables to be optimised, two associated with drought restriction rules and one with reservoir capacity. An objective function was formulated using reservoir capital costs and economic penalties for water shortages.
Probabilistic search methods are well suited to cope with the piecewise flat objective function surface characteristic of the headworks optimisation problem. This is because they have the inherent capability of exploring beyond the extent of a flat region. The efficiency and robustness of two probabilistic methods, the genetic algorithm and the shuffled complex evolution method, were investigated.

For the GA to be robust two lesser used genetic operators, inversion and population selection strategy, had to be used. These operators allowed the GA to employ a crossover rate close to 1 thereby enabling vigorous exploration of the objective function surface and hence avoiding premature convergence on a flat region. The near-optimal GA parameter settings can be used as a guideline for related problems.

The SCE method had comparable robustness but required (up to 25%) fewer evaluations than the GA. Both methods are considered applicable to such complex problems with stochastic forcing. Nonetheless, it was concluded that the GA was the preferred method. Optimisation of large urban headworks systems is so computationally demanding that parallel computing must be used. Whereas parallelism is inherent in the GA, the SCE method offers limited opportunity for parallel computing. Chapter 5 will consider how to implement parallel GA optimisation.
CHAPTER 5

APPROACHES FOR REDUCING COMPUTATIONAL TURNAROUND TIME
5.1 INTRODUCTION

Chapter 4 investigated two optimisation methods using a simplified hypothetical case study. It was demonstrated that modern probabilistic search methods such as the GA and SCE methods are typically robust and powerful in identifying a near-optimal solution. Nonetheless, it was noted that the excessive computational time requirement will continue to be a serious impediment in applying probabilistic search methods to complex water supply headworks systems. As a consequence, more powerful computing resources are needed; hence the interest is to explore the parallel implementation of GA and indeed other methods for improving computational efficiency for optimising complex stochastic water resources systems.

In this chapter, two approaches for reducing computational turnaround time will be explored. The first one is to implement a GA-based optimisation model using the Parallel Virtual Machines (PVM) protocol (Geist et al. 1994) as the inter-processor message-passing tool. The second approach introduces a simple method to further improve the efficiency of the Monte Carlo simulation. The method, referred to as replicate compression, can very substantially speed up Monte Carlo simulations for high reliability multi-reservoir systems.

5.2 THE NEED FOR PARALLEL COMPUTING

Parallel computing (several processors working together to solve the same problem) has become of paramount importance in several industrial applications, especially when the solution of large and complex problems must be found in a reduced time. Of relevance here is that parallel computing can significantly improve the turnaround time for probabilistic search methods that are capable of parallel implementation.

There are very few studies reporting the application of parallel GAs to water resources problems. As discussed in Chapter 4, the GA has a distinct advantage over the SCE method in parallel applications. The power of the GA lies in the implicit parallelism inherent in the processing of the strings (Goldberg 1989). Of significance
here is that each design is selected independently of the other designs in the current generation. Therefore the objective function for each design can be evaluated independently in each generation in a parallel-computing environment.

To appreciate the significance of this inherent parallelism, it is noted that the GA typically requires thousands of objective function evaluations to achieve a near optimal solution. If a GA searches through 20 generations and if each generation has a population of 50, the model has to evaluate the objective function 1000 times! Monte Carlo simulation is usually employed to incorporate stochastic climatic and other environmental inputs of the headworks system. Such simulations usually require a large number of replicates, and hence impose a considerable computational requirement. To appreciate the magnitude of the computational effort, a 1000-replicate simulation for the 9-reservoir Sydney water supply headworks system takes about 310 CPU seconds on a 750 MHz Pentium III processor. Therefore, a total CPU time of about 86 hours would be required to evaluate an objective function 1000 times. Clearly such CPU times severely limit the practical value of the GA method for optimising water supply systems.

In the following sections, the GA formulation described in Chapter 3 is modified to operate in a parallel-computing environment. This will enable a very significant reduction in turnaround time for an optimisation problem. If N processors are available, the turnaround time will be approximately 1/N of the single processor time.

5.3 PARALLEL COMPUTATION TOOL

In the parallel computation community, there are two important software protocols, Message Passing Interface (Pacheco 1996) and Parallel Virtual Machine (Geist et al. 1994). Message Passing Interface (MPI) is the more popular of the two. It is a complex and multifaceted system which can solve a wide range of problems. However, Parallel Virtual Machine (PVM) is equally capable and moreover, it is not difficult to migrate a PVM-based program to MPI. Because MPI and PVM are similar in capability and because the PVM facility was already available to the
author, this study will only consider the use of PVM to explore parallel GA optimisation of complex water supply headworks systems.

5.3.1 What is PVM?

PVM is a message-passing system that enables heterogeneous distributed computing. It allows a user to create and access a parallel computing system from a collection of distributed processors, and treat the resulting system as a single virtual machine (hence the name, parallel virtual machine). It was developed at Oak Ridge National Laboratory (Giest et al. 1994). The hardware in a user's virtual machine may be single processor workstations, vector machines or parallel supercomputers or any combination of those. The individual elements may all be of a single type (homogeneous), all different (heterogeneous) or any mixtures, as long as all machines used are connected through one or more networks. PVM is based on the message-passing model of parallel programming. Messages are passed between tasks over the connecting networks.

PVM can be used at several levels. At the highest level, e.g., the transparent mode, tasks are automatically executed on the most appropriate computer. In the architecture-dependent mode, the user specifies which type of computer is to run a particular task. In the low-level mode, the user specifies a particular computer to execute a task. In all of these modes, PVM takes care of necessary data conversions from computer to computer as well as low-level communication issues.

The PVM computing model is simple yet very general. It can program different components in different languages. The programming interface is deliberately straightforward, thus permitting simple program structures to be implemented in an intuitive manner. The user's application "decides" where/when its components are executed and determines its own control and dependency structure. Users may optionally control the execution location of specific application components. Tasks access PVM resources through a library of standard interface routines. These routines allow the initiation and termination of tasks across the network as well as communication and synchronization between tasks. The PVM computing model
accommodates a variety of application program structures. Because of its simple but complete programming interface, the PVM system has gained widespread acceptance in the high-performance scientific computing community (Adeli and Kumar 1995, Tsai et al 1999, Alonso et al 2000).

### 5.3.2 PVM Components

The PVM system is comprised of two main components. The first part is a daemon (or memory resident program) called `pvmd3` that resides on all the computers making up the virtual machine. The daemon oversees the operation of the user’s processes within a PVM application and coordinates inter-machine PVM communications. Each machine has one daemon maintaining a table of configuration and process information pertinent to the parallel virtual machine. User processes communicate with each other through the daemons. They first talk to their local daemon via the library interface routines. The local daemon then sends/receives messages to/from remote host daemons. When a user runs a PVM application, he first executes `pvmd3` on one of the machines (typically the login machine). This daemon in turn, starts up a PVM daemon on behalf of that user on all of the other computers in the user defined parallel virtual machine. The user may then execute his application.

The second part of the PVM system is the libraries of PVM interface routines. They contain the message passing routines that the application programmer may imbed in application code. The three PVM libraries are:

- **Libpvm3.a** - Library of C language interface routines. Always required.
- **Libfpvm3.a** - additionally required for Fortran codes
- **Libgpvm3.a** - required for use with dynamic groups.

The PVM libraries provide the ability to initiate and terminate processes; to pack, send, receive and broadcast messages; to synchronise via barriers; and to query and dynamically change the configuration of the parallel virtual machine. Application
programs must be linked with the appropriate PVM libraries. The PVM library routines do not directly communicate to other processes. Instead, they send commands to the local daemon and receive status information back. PVM terminology is presented in Appendix 1 and some of the Fortran routines is listed Appendix 2.

5.3.3 PVM Programming Structure

PVM can be programmed in four different approaches depending on the problem to be solved. The choice of model is application dependent and should be selected to best match the natural structure of the parallelized problem (Giest et al. 1994). It is beyond the scope of this study to examine in detail these four approaches. Instead, this section will outline the principal differences between the approaches with the aim of selecting the most suitable approach for this study.

**Master-Slave Model:** In this approach, a separate control program called the Master is responsible for process spawning, initialisation, collection and display of results. The Slave programs perform the actual computation involved in solving the problem on different processors (machines). Their workloads are allocated by the Master statically or dynamically. Each Slave process runs the same program. This is a traditional programming paradigm that has seen extensive applications (Tsai et al. 1999).

**Node-Only or One-Program Model:** In this approach, multiple instances of a single program execute on each processor (machine) available in the configuration. It is the responsibility of the programmer to determine which processor will be responsible to produce the final result. The programmer is responsible to make the load allocation among the processors approximately equal.

**Tree Computation model:** In this scenario, processes are usually spawned dynamically in a tree-like manner as the computation progresses, thereby establishing a tree-like, parent-child relationship as opposed to the Master-Slave applications where a star-like relations exists. This paradigm is less commonly used, but naturally fits applications where the total workload is not known a priori. The
branch-and bound, alpha-beta search, and recursive divide-and-conquer algorithms are a few examples in which this model of computation can be used.

**Hybrid Computational Model:** This model is a combination of the Tree model and the Master-Slave or One-Node models. This model possesses an arbitrary spawning structure. It takes the form of a general graph structure at any point during the computation.

The Master-Slave model is the most natural model for GA parallelisation. In a parallel-computing environment, the Master is responsible for process spawning, the GA initialisation; producing the new generation using GA operators (tournament, crossover and mutation, etc.); data collection from each processor for the whole population, and reporting of results. The Slaves perform the actual Monte Carlo evaluation of the objective function for each decision vector. Accordingly, the Master-Slave approach is adopted in this study.

### 5.3.4 PVM Load Balancing

Load balancing is very important for PVM applications. Making sure that each host is doing its fair share of work can significantly enhance performance. The simplest method is *static* load balancing. With static loading, each processor’s workload is fixed; the process knows a priori its share of the workload. In this method, the problem is divided up and tasks are assigned to processors only once. The partitioning may occur before the job starts, or as an early step in the application. The size and number of tasks can be varied depending on the processing power of a given machine. On a lightly loaded network, this scheme can be quite effective. However, it might encounter one or more processors that complete their portion of the workload much faster or much slower than the others. This situation could also arise when the machines in a PVM system are heterogeneous with different CPU speeds and memory or other system attributes.

When computational loads vary, a more sophisticated *dynamic* method of load balancing is required. The most popular method is called the *Pool of Tasks* paradigm. This is typically implemented as a Master/Slave program where the Master manages
a set of tasks. The Master sends Slaves jobs to do as they become idle. This method is not suited for applications that require task-to-task communication, since tasks will start and stop at arbitrary times. In this case, a third method may be used. At some predetermined time, all the processes stop; the workloads are then re-examined and redistributed as needed. Variations of these methods are possible for specific applications. The next section will discuss the Pool of Tasks method that is used in this study.

5.4 PARALLEL GA IMPLEMENTATION

This section describes how the GA is implemented in a parallel computing environment. Figure 5.1 outlines the parallel GA implementation using the Master-Slave approach. In the Master program, the PVM process is spawned and the number of available processes is identified. The GA starts to generate the initial population of decision variables, the decision variables in the population are sent from the Master to all Slaves. Each Slave processor may then evaluate the objective function based on the decision variables it receives from the Master processor. After the initial population is evaluated, the Master enters the main GA iteration loop. It produces the next generation of decision variables using genetic operators such as tournament selection, crossover, mutation, inversion, etc. PVM routines are then employed to send the decision variables to idle Slaves to evaluate the objective function. The Master then waits and receives the results from the Slaves. When the results for the whole population are received, the Master will apply the PSS strategy to the population; check against the GA stopping criteria; and save the results. Depending on the stopping criteria, the Master will either repeat the GA iteration or terminate the program.

In the Slave program, each Slave is set to be in an infinite loop. It waits until it receives a decision vector sent from the Master. Then the objective function is evaluated and the result is sent back to the Master.
The Master program

Spawn the PVM
Generate initial population
Send initial decision vectors to the Slaves
Evaluate the initial objective functions.

Do GA iteration

- Produce a new generation of decision variables via tournament selection, crossover, mutation, and inversion operations
- Send decision vectors to idle Slave processors
- Wait and receive objective function value from any Slave processor
- Implement population selection strategy
- Terminate GA if stopping criterion is met

End GA iteration

- Display results
- Kill Slaves
- End GA program

The Slave program

Do

- Wait for message with decision vectors
- Evaluate objective function
- Send objective function value to Master

End do

Figure 5.1 A Master-Slave parallel application of the genetic algorithm
Note that the execution time required to evaluate the same objective function may be different from one processor to another, possibly because there is a heterogeneous collection of machines in the virtual machine or time sharing constraints. In such cases, one has to consider the effects of having different computational power on each processor in the configuration. If tasks are evenly distributed to Slave processors, the application will run as slow as the task on the slowest processor.

This problem can be largely avoided by incorporating some form of load balancing into the parallel application. In this study, a dynamic load balance scheme, *Pool of Tasks*, is maintained among processors. Figure 5.2 illustrates the *Pool of Tasks* load balancing used in the GA parallel implementation. For each generation in the GA search, after the new population is generated, each Slave processor is only assigned with one set of decision variables at a time for objective function evaluation. The Master then enters in a ‘busy-wait’ loop looking constantly for an idle Slave to which a new task is assigned or for the arrival of the objective function value from a Slave process. With this method, all the Slave processors are kept busy as long as there are unfinished tasks.

**5.5  REPLICATE COMPRESSION**

As noted in the Introduction, this chapter aims to explore two approaches to improve computational turnaround efficiency. Section 5.2 to 5.4 outlined the first approach of using parallel computing technique to reduce the computational turnaround time. In this section, the second approach will be introduced to improve the efficiency of the Monte Carlo simulation itself.

**5.5.1 Rationale For Replicate Compression**

In the case study described in Chapter 4, the restriction rules were optimised for an urban water supply system with a single reservoir. If realistic optimisation of operating rules is to be realised, there appears to be little choice but to use Monte Carlo simulation. It was found that the evaluation of the objective function using the Monte Carlo simulation model consumes almost all the processor time in a GA optimisation, and a large number of replicates are required for statistically
meaningful risk estimates of system performance. The excessive computational time requirement was a serious impediment in applying optimisation to large-scale complex water supply headworks systems. Therefore, it is worthwhile exploring any opportunity to reduce the simulation burden.

![Define variables](image)

---

The excess of computational time requirement was a serious impediment in applying optimisation to large-scale complex water supply headworks systems. Therefore, it is worthwhile exploring any opportunity to reduce the simulation burden.

---

**Figure 5.2 Pseudo-code for Pool of Task load balancing**

---

Figure 5.3 illustrates how the restriction penalty component of the objective function is evaluated in the Monte Carlo simulation model. The penalty values are zero for the time period when the storage levels are above the restriction trigger level. A penalty is imposed only when the system storage falls below the trigger level (the shaded area). In fact, for systems with high reliability, most replicates have few, if any, significant drawdowns. To evaluate drought security (in terms of probability...
of restrictions and associated penalties), it suffices to simulate the time periods with significant drawdowns when the storage level is below the trigger level. Therefore, if the objective function only depends on restriction penalties, it needs to be only evaluated for the replicates which experience one or more periods where system storage falls below the trigger level. This can very significantly reduce the simulation effort particularly if the system is designed to operate with a high level of reliability. This approach is called replicate compression.

5.5.2 Identification of Critical Period

Replicate compression is based on the idea of using critical periods only. McMahon and Mein (1987) defined the critical period as a period during which a reservoir goes from a full condition to an empty condition without spilling in the intervening period. The start of a critical period is a full reservoir; the end of the

Figure 5.3 Schematic illustrating how restriction penalty is imposed
critical period is when the reservoir first empties. Thus only one drought is allowed to occur during a critical period. Figure 5.4 illustrates the concept.

However, this definition is not universally accepted. For example, another definition is that the critical period is the period during which a reservoir goes from a full condition through emptiness to the full condition again without spilling in the intervening period (McMahon and Mein 1987). In this study, the second definition is adapted to describe the replicate compression method. Returning to Figure 5.3, since the penalty is only imposed when the system storage level falls below the restriction trigger level, the critical period is redefined as follows. The critical period commences when a full reservoir starts a drawdown which passes through the restriction trigger level and ends when the reservoir storage passes through the trigger level during its recovery. Figure 5.3 illustrates this concept.

5.5.3 The Replicate Compression (RC) Method

The replicate compression method seeks to identify the critical period(s) for each replicate. Starting with a full reservoir it simulates each critical period and evaluates

![Image removed for copyright reasons]

The full thesis is available in the Auchmuty Library

Figure 5.4 Example of critical periods: Mitta Mitta River (from McMahon and Mein 1987)
restriction penalties.

Figure 5.5 presents the pseudo code to implement the RC method. For the $j^{th}$ replicate, the following steps are executed in the simulation model:

1. Lump the total streamflow inflow($i, j$), total reservoir capacity lumpCap($i$), total reservoir evaporation evap($i$), and total unrestricted demand demand($i$), where $i$ is the simulation time step.

2. Lump the storage in the multiple reservoirs into a single storage whose volume at the end of time step $i$ is

   \[
   \text{storage}(i) = \min\{\text{lumpCap}(i), \max[0, \text{storage}(i-1) + \text{inflow}(i, j) - \text{evap}(i) - \text{demand}(i)]\}
   \]

3. For the entire simulation time period, identify all the time steps at which the lumped storage storage($i$) is below the trigger level trigger($j$) $\times$ lumpCap($i$), where trigger($j$) is the trigger storage fraction for the $j^{th}$ replicate. Record these steps as emptStep($k$), $k = 1, \ldots, n\text{Empty}$.

4. Identify all the time steps at which the reservoir is in a full condition. Record these steps as fullStep($k$), $k = 1, \ldots, n\text{Full}$.

5. Identify the critical periods. A critical period commences at step StartCrit($n$), $n = 1, \ldots, n\text{Crit}$ when the lumped system is full. The total storage must drop below the storage at which restrictions are triggered. The critical period ends at step EndCrit($n$), $n = 1, \ldots, n\text{Crit}$, when the lumped storage is full again.

6. For each critical period, start at the end of the critical period and work backwards in time, until the storage drops below the trigger level. Reset EndCrit to minimise the length of the critical period requiring simulation.
Do  j  = 1, number of replicates
   nFull = 0
   nEmpty = 0
   nCrit = 0
   Do  i = step1, step2  !simulation time
      !Find all the steps which the storage is below the triggered level
      Storage(i) = max[0, storage(i-1)+inflow(i,j)-evap(i)-demand(i)]
      If  storage(i) ≤ trigger(j)×max Cap(i)
         nEmpty= nEmpty + 1
         EmptStep(nEmpty) = i  ! Record these steps in order)
      Else if storage(i) ≥ maxCap(i)
      !Find all the steps for which the storage is full
         nFull= nFull + 1
         fullStep(nFull) = i  ! Record these full steps in order
         Storage(i) = lumpCap(i)
      Endif
   End do
   ! Identify the first critical period. Consideration is given to the case that a shortfall before
   the reservoir is full
   nCrit = 0
   If nEmpty = 0 Return
      position =0
      Call Y_in_X( emptStep(1), fullStep, numFull, position, newPosition)
      nCrit = nCrit +1
   If   newPosition > numFull
      endCrit(nCrit) = step2
   Else
      endCrit(nCrit) = fullStep(newPosition)
   Endif
   If newPosition =1  ! Reservoir not full before the first shortfall
      startCrit(nCrit) = step1
   Else  ! Reservoir was full before the first shortfall
      StartCrit(nCrit) = fullStep(newPosition-1)
   End if
   position = newPosition
!Work backwards till storage drops below trigger level – to reset the end of
   critical period.
   Call reset_endCrit_point
! Identify the remaining critical period(s)
   Do k = 2, nEmpty
      Call Y_in_X( emptStep(k), fullStep, numFull, position, newPosition)
      If (newPosition > position) Then
         nCrit = nCrit + 1
         If (newPosition > numFull) Then
            endCrit(nCrit) = step2
         Else
            endCrit(nCrit) = fullStep(newPosition)
         End if
      End if
      If (newPosition =1) Then
         startCrit(nCrit) = step1
      End if
   Else
The Monte Carlo simulation model is then performed for each critical period. The assumption is made that because the lumped system is full at the start of the critical period, there is a high probability that all the reservoirs in the system are full or close to full. The assumption is expected to be satisfactory in most urban headworks systems, where storage is balanced in proportion to reservoir capacity.

The replicate compression method has been implemented in the WATHNET (Kuczera 1992) simulation model. Figure 5.6 presents an example using replicate compression for a 600-month time period in the WATHNET simulation model. In this example, three critical periods are identified and simulated. The shaded time periods are skipped, thereby reducing the computational effort for the replicate.

Returning to the Sydney headworks example mentioned earlier, if no replicate compression were used, a GA executed on a 750 MHz Pentium III processor would
require a total CPU time of about 86 hours to perform 1000 function evaluations. By combining replicate compression with PVM using 10 processors with an efficiency of 99%, the turnaround time for performing a GA optimisation of the Sydney headworks system can be reduced to 8.5 hours. This turnaround time can be further reduced because 1.2 GHz Athlon-based processors will be used in parallel GA application.

Usually urban water headworks systems are designed so that droughts causing restrictions occur about once every 10 to 20 years, therefore replicate compression is expected to substantially reduce the turnaround time. The efficiency of replicate compression will be assessed and discussed in Chapter 8.

![Figure 5.6 Illustration of replicate compression using WATHNET](image.png)

5.6 CONCLUDING REMARKS

This chapter discussed two approaches for reducing the computational turnaround time for GA optimisation. The GA optimisation scheme was discussed in detail in Chapter 3 and formulated and demonstrated with a simple case study in Chapter 4. In
this Chapter, the GA scheme was modified to operate in a parallel computing environment. Therefore, instead of using one processor to evaluate the objective function, multiple processors can be used to conduct the GA objective function evaluation simultaneously. The use of parallel computing will dramatically reduce the turnaround time, which makes the optimisation of complex water supply systems practically possible. The second approach to reduce turnaround time was based on replicate compression which reduces actual simulation effort by identifying critical periods during which restrictions are to be encountered. However, a limitation of replicate compression is that it can only be used if the objective function depends on penalties arising from restrictions. These approaches will be applied to a case study involving the Sydney headworks system, the subject of the next three chapters.
CHAPTER 6

CASE STUDY: SYDNEY
HEADWORKS SYSTEM
6.1 INTRODUCTION

Starting with this chapter, a case study involving the Sydney water supply headworks system is presented to demonstrate the parallel GA and to highlight problems encountered with its application.

Figure 6.1 presents the general framework for optimising the Sydney headworks system. The optimisation procedure starts from an initial trial solution generated by the GA and proceeds towards a near-optimal solution in a sequential manner. During the search, decision variables (presented in Chapter 7) are passed from the GA to the

Figure 6.1 Schematic showing interaction between genetic algorithm and simulation model of the headworks system
simulation model to perform the Monte Carlo analysis. WATHNET, a software package developed by Kuczera (1997), is used to perform the headworks simulation, following which the objective function is evaluated. Two groups of stochastic input data are required to perform the Monte Carlo analysis, i.e. multiple replicates of hydroclimate data and multiple replicates of future demand based on population and climate. Based on the new values of the objective function, the GA trials a new decision vector. This is repeated until there is no further significant improvement in objective function.

The GA and its parallel implementation have been described in Chapter 5. This chapter begins with a description of the WATHNET simulation model. It then describes how the WATHNET is applied to the Sydney headworks system. It is stressed that because the primary purpose of the case study is to demonstrate the parallel GA judicious use of simplifying assumptions has been employed.

6.2 DESCRIPTION OF WATHNET SIMULATION MODEL

6.2.1 WATHNET Overview

Generalised simulation models such as HEC-3 (HEC 1981), HEC-5 (HEC 1982) and more recently IRIS (Loucks and Salewicz 1989) use explicit rules to make water assignments such as reservoir releases and link allocations. Many water supply systems are operated in practice using such rules (Loucks and Sigvaldson 1982). Explicit-rule simulation models require detailed specification of operating rules because no optimising algorithm is used to make assignments. As the size of the system grows, definitions of these rules can become onerous. For dynamic simulations in which average demand and/or system configuration change over time, the complexity of the rules can dramatically increase.

WATHNET is an example of a generalised simulation model that departs from the traditional approach to system operation. It uses a network linear program (NetLP) to simulate the operation of a wide range of water supply headworks configurations. Instead of using explicit rules to make water assignments, it uses information about the current state of the system as well as forecasts of streamflow and demand to
formulate a network linear program. In a single time step, the NetLP determines the water allocation for given streamflow and demand in accordance with the following hierarchy of objectives:

1. Satisfy demand at all demand zones (if there is insufficient water in the system, or if there are transfer limitations, rationing will occur);
2. Satisfy all instream flow requirements;
3. Ensure that reservoirs are at their end-of-season target volumes;
4. Minimise delivery costs;
5. Avoid unnecessary spill from the system.

The network linear program used in WATHNET can be formulated in two modes:

**Single time step simulation:** In this mode, streamflow and demand forecasts for the current time step are used in conjunction with the reservoir levels at the start of the time step to simulate the allocation of water within the headworks system.

**Multiple time step simulation:** Perfect knowledge of streamflow and demand over a period involving multiple time steps is used by the network linear program to allocate water. This allows the linear program to anticipate future drought and if physically possible, cushion the system against the drought.

The second mode is not considered realistic for the purpose of simulating headworks operation. It uses information unavailable to operators and hence exaggerates the drought security of the system. For this reason, the single time step simulation mode has been adopted in all the WATHNET headworks simulations reported in this study.

### 6.2.2 Network Linear Programs

A NetLP is a linear program which finds the minimum cost solution for conveying a commodity (water in this study) through a network of unidirectional arcs interconnecting supply, demand and transhipment nodes. Kennington and Helgasen (1980) provided a detailed treatment of network linear programming.

The easiest way to understand how WATHNET formulates the NetLP to solve the seasonal water assignment problem is to illustrate it for a simple water supply
system. Figure 6.2 (Kuczera 1997) depicts a headworks system consisting of two reservoirs with active capacities $\text{CAP}_i$, start-of-season volumes $\text{INIT}_i$, and inflow volumes $\text{I}_i$, $i = 1, 2$. These reservoirs deliver water through a conduit network to two demand zones with demand volumes $\text{D}_i$, $i = 1, 2$. Water transferred from reservoir 2 to demand zone 1 must be pumped at a unit cost of 10. All remaining conduits incur minimal delivery cost. Spills from the reservoirs flow downstream channels to a waste node. For both streams there is an instream flow requirement of $Q_{MIN}$.

**Figure 6.2 Sample water supply headworks system (from Kuczera 1997)**

Figure 6.3 displays a NetLP graph which will operate the system according to the hierarchy of objectives (Kuczera 1997). Comparing Figures 6.2 and 6.3, Figure 6.3 has an extra node as well as many more arcs. They have been added so that the water assignments made by the NetLP are consistent with the operating objectives. The additional node in Figure 6.3 is called the balancing node. It balances the system in the sense that the sum of nodal inflows equals the sum of nodal outflows.

$$RB = \text{INIT}1 + \text{I}1 + \text{INIT}2 + \text{I}2 - \text{D}1 - \text{D}2 \quad (6.1)$$

The balancing node collects streamflow that flows to node 2, the waste node. This is done by introducing a spill arc which flows from node 2 to node 1. The spill arc has zero penalty and is uncapacitated indicating its minimum and maximum
capacities are 0 and $\infty$ respectively; this information is summarised by the triplet (0, 0, $\infty$), where the first element is the cost per unit volume and the remaining two are the minimum and maximum volumes that can be carried by the spill arc. In addition, the balancing node is connected to the shortfall and storage carryover subnetworks. All the subnetworks in Figure 6.3 have two arcs sharing the same from- and to-nodes.

Figure 6.3 Graph of NetLP for sample water supply headworks system shown in Figure 6.2

The shortfall subnetwork consists of two arcs flowing from the balancing node to a demand node. These arcs carry the highest penalties in the NetLP and, therefore, would be utilised only if demand could not be satisfied by any other physically possible assignment. The introduction of the shortfall arcs forces the NetLP to satisfy demand, where possible, as its the highest-priority objective. If the shortfall subnetwork is employed, it is physically impossible to meet demand. In such a situation rationing would occur. Without the shortfall subnetwork the NetLP would return an infeasible solution in the event demand could not be satisfied. WATHNET
allows between 1 and 9 shortfall arcs to be assigned to each demand node in order to provide a mechanism for sharing the burden of demand shortfalls evenly over all demand nodes having the same supply priority.

The instream subnetwork is introduced for every stream arc that has a non-zero instream flow requirement. The subnetwork employs two arcs. The first arc has the parameter triplet (-1000000, 0, QMIN), while the second has (0, 0, ∞). Because the first arc has a negative penalty, the NetLP will try to fully utilise it to minimise the objective function. The absolute value of the penalty is less than the shortfall penalties, but greater than any other penalties in the NetLP. Therefore, the NetLP will make satisfying instream flow requirements its second highest priority.

The storage carryover subnetwork consists of arcs flowing from reservoir nodes to the balancing node, which can be thought of as a sink collecting all streamflow out of the system as well as water carried over to the next season – the NetLP cannot store water at nodes. Consider the reservoir at node 5 in Figure 6.3, the seasonal mass balance for the reservoir is given by

\[ \text{Initial storage } + \text{Inflow} = \text{Release to demand nodes 3 and 6 } + \]

\[ \text{Spill to waste node 2 } + \text{Final storage} \]  \hspace{1cm} (6.2)

The initial storage and inflow are represented by the nodal requirement. At least two carryover arcs are required for each reservoir. The first arc, the target arc, encourages the NetLP to store, if possible, water in the reservoir up to its target volume. This is accomplished by setting a negative penalty whose absolute value is selected so that it is only exceeded by shortfall and instream penalties. The above-target arc conveys storage in excess of the target volume to the balancing node. However, because it has a small positive penalty, the NetLP has no incentive to store in excess of the target volume. WATHNET allows between 1 and 20 target arcs to be assigned to each reservoir node to balance the volumes of reservoirs having the same filling priority. Thus reservoirs with filling priority 1 will be filled in preference to reservoirs with a lower filling priority; and, conversely, reservoirs with
a priority greater than 1 will be drawn down in preference to reservoirs with priority 1. It is noted that the user can override this hierarchy by assigning custom penalties to the subnetworks.

### 6.2.3 Example of Network Operation

This section illustrates how the NetLP works. Figure 6.4 depicts the system during a drought and displays the assignments made by the NetLP. Table 6-1 lists the arc parameters for all the arcs in Figure 6.4. Reservoirs 1 and 2 have capacities of 600 and 350 units respectively, end-of-session target volumes of 500 and 300 respectively, and both have a filling priority of 1. Both demand nodes have the same supply priority, namely 1.

Figure 6.4 Example: demonstrating a demand shortfall due to water shortage
Table 6-1 Arc parameters for network in Figure 6.4

<table>
<thead>
<tr>
<th>Arc type</th>
<th>Node</th>
<th>Penalty</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>From</td>
<td>To</td>
<td>Minimum</td>
</tr>
<tr>
<td>conduit</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>conduit</td>
<td>5</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>conduit</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>stream</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>stream</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>spill</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>shortfall</td>
<td>1</td>
<td>3</td>
<td>11000000</td>
</tr>
<tr>
<td>shortfall</td>
<td>1</td>
<td>3</td>
<td>11010000</td>
</tr>
<tr>
<td>shortfall</td>
<td>1</td>
<td>6</td>
<td>11000000</td>
</tr>
<tr>
<td>shortfall</td>
<td>1</td>
<td>6</td>
<td>11010000</td>
</tr>
<tr>
<td>instream</td>
<td>4</td>
<td>2</td>
<td>-500000</td>
</tr>
<tr>
<td>instream</td>
<td>5</td>
<td>2</td>
<td>-500000</td>
</tr>
<tr>
<td>carryover</td>
<td>4</td>
<td>1</td>
<td>-3000</td>
</tr>
<tr>
<td>carryover</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>carryover</td>
<td>5</td>
<td>1</td>
<td>-3000</td>
</tr>
<tr>
<td>carryover</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The initial storage and inflow to reservoirs 1 and 2 are 70 and 50 units respectively. Demands $D_1$ and $D_2$ are 150 and 100 units respectively. Clearly there is insufficient water to meet demand. Therefore, an inflow of 130 units is required at the balancing node to balance the system. This inflow is conveyed through the shortfall arcs to ensure a mass balance at nodes 3 and 6, resulting in a demand shortfall of 80 and 50 units respectively. Observe that the instream flow requirements were not satisfied because the higher shortfall penalties forced the NetLP to divert all available water to the demand nodes. Also note that both reservoirs have zero flow in their carryover sub-networks meaning their end-of-season volumes are zero.

6.2.4 Description of The WATHNET Program

WATHNET is made up of five programs. Figure 6.5 is a schematic showing the logical relationship between these programs and the files required to run each program:

EDNET: This program defines and edits the headworks network. The network schematic is drawn using a graphical user interface. For each arc or node created, necessary data is entered via forms. EDNET creates a network file that is input to SIMNET.

WATSTRM: This program manages data files for streamflow, climate and demand variables at multiple locations. It also can be used to stochastically generate multiple replicates of future variables at nominated sites. The stochastic model uses a multi-site lag-one Markov model to generate annual time series and then uses a nonparametric method, the method of fragments, to disaggregate the variables to the required time step. The output from WATSTRM is input to SIMNET.

SIMNET: SIMNET is the heart of WATHNET. Using the files provided by EDNET and WATSTRM along with run options entered via a graphical user interface, SIMNET formulates the NetLP, solves it and saves relevant results to a network output file.
WATOUT: SIMNET creates an unformatted network output file, which stores in compressed form all the simulation results. WATOUT is used to study the contents of this file. Using a graphical user interface, extensive graphical and tabular reporting of results can be performed.

SIMBAT: SIMBAT is a batch implementation of SIMNET designed to run in a console environment. It reads a batch file created by SIMNET and then proceeds with simulation/optimisation analysis. It allows big jobs to run in the background without the need for a graphical interface. In this case study, the batch file is created in SIMNET for use by SIMBAT.

Image removed for copyright reasons

The full thesis is available in the Auchmuty Library

Figure 6.5 Schematic of WATHNET programs (From Kuczera 1997)
6.2.5 Input Data – Streamflow Generation

WATHNET requires streamflow (and optionally other climatic data, reservoir evaporation and demand) data to be stored on a file with a special format. This file is called a WATHNET streamflow file. The program WATSTRM can be used to simplify the creation and manipulation of such streamflow files and to generate multiple replicates of input. A detailed description of generating streamflow can be found in WATHNET manual (Kuczera 1997). This section will only provide a brief outline of this procedure.

Figure 6.6 shows in a flowchart the procedure to generate streamflow data. First WATSTRM fits a multi-site probability model to annual streamflow data. From the best-fit estimates of the probability model parameters are obtained as well as their distributions which describe the uncertainty about the parameters. For each replicate, a check is made whether parameter uncertainty is to be allowed for. If so, the probability model parameters are randomly sampled from their distributions; otherwise, the best-fit estimates are used. Then a specified number of years of data are generated using the probability model and disaggregated into monthly flows using the method of fragments.

This procedure is repeated until the desired number of replicates has been generated. Each replicate represents an equally likely sequence of future streamflows. SIMNET simulates system operation for each of these replicates, and then WATOUT provides a statistical summary of system performance.

6.2.6 Input Data – Demand Generation

Demand for water may be correlated with climate. WATHNET assumes that the user deals with this issue after generating replicates of hydroclimate data using WATSTRM. The issues dealing with Sydney case study will be deferred till Section 6.3.4.
Fit probability model to historical annual data

Parameter estimates and their distributions

$k = 1$

Allow for parameter uncertainty?

Yes

Randomly sample annual probability model parameters

Generate annual flows for replicate $k$

Disaggregate into seasonal flows

$k = k + 1$

No

Is $k$ large enough?

Yes

Stop

No

Use best parameter estimates

Figure 6.6 Flowchart of generating seasonal streamflow data
6.3 APPLICATION OF WATHNET TO THE SYDNEY HEADWORKS SYSTEM

This section describes the application of the WATHNET simulation model to the Sydney headworks system.

6.3.1 Overview of The Sydney Headworks System

The Sydney water supply headworks system presently provides water to over four million people over an area of 13,000 km$^2$. The population served by the system is growing at a rate of about 3% per annum (ABS 1998). The total metered consumption is presently over 500,000 ML per annum with the water supplied from 9 major reservoirs. To assure a reliable water supply for Sydney, it is necessary to store up to 64,000 litres per person, which represents one of the highest amounts of stored water among the larger cities in the world. Operation of a water supply system in such conditions represents an ongoing challenge to the Sydney Catchment Authority to provide a reliable water supply at minimum cost.

The schematic diagram of the Sydney water supply headworks system is shown in Figure 6.7. The water supply is drawn from four main river systems: Upper Nepean, Warragamba, Shoalhaven and Woronora. Warragamba reservoir is the largest and the most important reservoir in the system, providing approximately 80% of the storage for the headworks system and receiving 74% of the average annual inflow (excluding the Shoalhaven River). Avon and Nepean reservoirs are the main source of water supply to the Illawarra region, with augmentation from the Shoalhaven via Nepean reservoir during droughts.

The Upper Nepean catchment is in the zone of the highest annual rainfall on the New South Wales central coast. Four reservoirs store and regulate the supply from this area – Cataract, Cordeaux, Avon and Nepean. Water released for supply of Sydney flows from Nepean and Cordeaux reservoirs through natural river channels to a diversion weir at Pheasants Nest. The diverted water is transported through the Nepean tunnel to Broughtons Pass weir where water released from Cararact reservoir...
Image removed for copyright reasons

The full thesis is available in the Auchmuty Library

Figure 6.7 Schematic of Sydney headworks system (SMEC 1996)
is also intercepted. After supplying the local demand, the remaining water is transferred to Sydney via a system of tunnels, aqueducts and open channels to Prospect reservoir. The transfer capacity of this system is 680 ML/day. Nepean and Avon reservoirs are connected via the Nepean-Avon tunnel with the transfer capacity depending on the difference between the reservoir water levels.

Water from Avon reservoir is mainly used for supply of the Illawarra region. Supply from Warragamba reservoir to Sydney is delivered by gravity to Prospect reservoir through pipelines with diameters ranging from 2650 to 3000mm. Prospect reservoir is the major distribution reservoir in the system. It receives water from both the Warragamba and Upper Nepean catchment areas. The maximum inflow capacity to Prospect reservoir is 2600 ML/day.

At Lake Yarrunga on the Shoalhaven River, water is lifted 612m to Fitzroy Falls reservoir using two pumping stations. From Fitzroy Falls reservoir water is transferred to Wingecarribee reservoir by canals and a pumping station, where it is stored and distributed either to Nepean reservoir or to Warragamba reservoir.

6.3.2 Review of Current Headworks Operating Policy

The current operating policy consists of a set of rules: the reservoir drawdown curves which specify how a given volume of water should be distributed across the reservoirs; the restriction trigger levels which implement imposition of restrictions on consumption; and a pump mark which defines the system storage below which pumping from the Shoalhaven system commences. The following subsections describe these operating rules in more detail.

6.3.2.1 Drawdown Curves

The operation of the Sydney water supply headworks system requires decisions to be made regarding the level to which storages shall be drawn down to meet a particular requirement. In terms of maximising the security of supply, the choice must be based on minimising spill from the system. Therefore, at any time water should be drawn from those storages which are most likely to spill. At present, the
priority for drawdown of reservoirs is based on a diagram which is indicative of the likelihood of spill from each reservoir shown in Figure 6.8 (SMEC 1996).

The drawdown curves can be incorporated into the WATHNET simulation model. The WATHNET package allows for drawdown rules by assigning a filling priority to each reservoir. Reservoirs with the same filling priority will tend to fill and empty together. This is achieved by dividing each storage into zones from top to bottom and assigning a cost to the water in each zone with the lowest cost at the top. In this way, provided that all other costs are equal, water from the top zone in each reservoir with a particular filling priority will be used before water from the second top zone in any reservoir and so on.

6.3.2.2 Restriction policies

Water restrictions are applied in many cities in Australia during droughts with the aim to reduce consumption when reservoir levels reach unacceptably low levels. The restriction policy usually specifies different levels of restrictions which are activated when storage falls below pre-defined levels. The restriction policy typically allows for progressively more severe restrictions to be applied as storage levels fall. Table 6-2 presents the restriction policy for the Sydney headworks system (SMEC 1996). It is primarily targeted at outdoor domestic water use.

The restriction policy used in this study has four levels:
Level 1: This level of restrictions is applied whenever the total storage falls to 65% of full storage capacity. By banning the use of fixed hoses and sprinklers, it is estimated that average daily consumption will be reduced by 15%.
Level 2: When the total storage falls to 55% of capacity, consumption is reduced by restricting the use of hand-held hoses to two hours per day
Level 3: When the total storage falls to 45% of capacity, the consumption is reduced by restricting the use of hand-held hoses to one hour per day.
Level 4: When the total storage falls to 30% of capacity, the use of water outside of dwellings is completely banned resulting in an expected reduction of average daily consumption equal to 40%.
Image removed for copyright reasons

The full thesis is available in the Auchmuty Library

Figure 6.8 Existing drawdown rules (from SMEC report: Water Supply Strategy Review - Phase II 1996)
The band of storage volumes selected for each level of restrictions is usually determined from simulation using long periods of synthetic streamflows to obtain the probability of restrictions. The storage level which triggers level 1 restriction is selected so the probability of experiencing restrictions in any year is 1/30.

Table 6-2 Sydney headworks restriction policy (SMEC report 1996)

<table>
<thead>
<tr>
<th>Restriction levels</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage triggered level (%)</td>
<td>65</td>
<td>55</td>
<td>45</td>
<td>30</td>
</tr>
<tr>
<td>Expected reduction in consumption (%)</td>
<td>15</td>
<td>20</td>
<td>30</td>
<td>40</td>
</tr>
</tbody>
</table>

WATHNET uses storage-triggered restriction rules. In this study, restrictions are only limited to outdoor residential demand, and there are no restrictions imposed on commercial or indoor water consumption. For the demand node belonging to outdoor demand nodes, the actual demand $AD$ that the NetLP attempts to satisfy is determined by the following equation:

$$AD = \begin{cases} 
D & \text{if } S \geq S_0 \\
D \left( \frac{100 - \Delta_i}{100} \right) & \text{if } S_{i-1} > S \geq S_i, i = 1,...,4 
\end{cases}$$

(6.3)

where $D$ is the unrestricted demand, $S$ is the percent fullness at the start of the season for a selected group of reservoirs, $S_i$ is the percent fullness at the bottom of restriction zone $i$ with $S_4$ always zero, and $\Delta_i$ is the percent reduction in unrestricted demand if $S$ lies in zone $i$. 
Figure 6.9 illustrates a demand restriction rule. Because the total storage falls in zone 1, the rule imposes a 10% reduction in the demand of all nodes belonging to the restriction group.

6.3.2.3 Shoalhaven Pumping Mark

During prolonged periods of less-than-average streamflows the Shoalhaven scheme is used to transfer what are essentially run-of-river flows from Tallowa Dam to the Nepean and Warragamba reservoirs to supplement the yield of those catchments. The pump mark is the storage level (expressed as percentage of the Warragamba storage) at which pumping from the Shoalhaven storage would commence. This rule was developed using historical streamflows and current demand so that one year of consumption would be held in storage at the critical point of the worst historical drought. As demand increases with time, the pump mark will have to increase with a consequent increase in pumping from the Shoalhaven system.

6.3.3 Network Schematic of Sydney Headworks System

The program EDNET was used to develop a network schematic of the Sydney headworks system and define the required information about each arc and node in the schematic. Figure 6.10 shows a schematic representation of the Sydney headworks...
system as drawn by EDNET. Table 6-3 describes the nodes presented in Figure 6.10, whereas Table 6-4 describes in detail the function of each node. The two arc types supported by WATHNET are described in Table 6-5.

Comparison of the system schematic in Figure 6.7 and the EDNET schematic in Figure 6.10 reveals that Figure 6.10 has extra nodes or arcs which have been added for this case study:

1. In the existing system, only one pump mark is used to distribute water to either Nepean or Warragamba reservoir (arc from node 36 to node 8). This pump mark is dependent on the Warragamba reservoir storage below which the water from the Shoalhaven system is transferred to the Sydney metropolitan demand area. Based on a preliminary study, it was found that using one pump mark for the Sydney headworks system may cause the following problem; namely that water restrictions may be imposed in the South Illawarra demand zone while there is still sufficient water supply in Warragamba reservoir. Therefore, another pump mark (arc from node 36 to node 7) is added in Figure 6.10. It depends on the Avon reservoir storage below which the water from the Shoalhaven River is transferred to the South Illawarra system.

Figure 6.11 illustrates how the pump mark for Warragamba reservoir is implemented in WATHNET. Assume that the pump mark for Warragamba reservoir is 51% (the right column in Figure 6.11). The maximum monthly volume that can be transferred from the Shoalhaven system is 64000 ML (100% of the arc capacity in the left column in Figure 6.11). If Warragamba is at or below 51% storage, the maximum arc capacity is enabled thereby allowing transfer. If Warragamba storage exceeds 52%, no water can be transferred.
Figure 6.10 EDNET schematic of Sydney headworks system
Table 6-3 Description of nodes appearing in the schematic shown in Figure 6.8  
(From Kuczera 1997)

<table>
<thead>
<tr>
<th>No</th>
<th>Node name</th>
<th>Node type</th>
<th>No</th>
<th>Node name</th>
<th>Node type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Warragamba Reservoir</td>
<td>R</td>
<td>19</td>
<td>Fitzroy Falls Inflow</td>
<td>S</td>
</tr>
<tr>
<td>2</td>
<td>Woronora Reservoir</td>
<td>R</td>
<td>20</td>
<td>W. Reef Inflow</td>
<td>S</td>
</tr>
<tr>
<td>3</td>
<td>South Coast Junction</td>
<td>J</td>
<td>21</td>
<td>Penrith Weir</td>
<td>S</td>
</tr>
<tr>
<td>4</td>
<td>Cataract Reservoir</td>
<td>R</td>
<td>22</td>
<td>Nepean Inflows</td>
<td>S</td>
</tr>
<tr>
<td>5</td>
<td>Cordeaux Reservoir</td>
<td>R</td>
<td>23</td>
<td>Avon Inflows</td>
<td>S</td>
</tr>
<tr>
<td>6</td>
<td>Avon Reservoir</td>
<td>R</td>
<td>24</td>
<td>Cordeaux Inflows</td>
<td>S</td>
</tr>
<tr>
<td>7</td>
<td>Nepean Reservoir</td>
<td>R</td>
<td>25</td>
<td>Cataract Inflows</td>
<td>S</td>
</tr>
<tr>
<td>8</td>
<td>Wingecarribee Reservoir</td>
<td>R</td>
<td>26</td>
<td>Woronora Inflows</td>
<td>S</td>
</tr>
<tr>
<td>9</td>
<td>Fitzroy Falls Reservoir</td>
<td>R</td>
<td>27</td>
<td>Wingecarribee Release</td>
<td>J</td>
</tr>
<tr>
<td>10</td>
<td>Lake Yarrunga</td>
<td>R</td>
<td>28</td>
<td>Sydney Commercial</td>
<td>D</td>
</tr>
<tr>
<td>11</td>
<td>Broughton Pass Weir</td>
<td>G</td>
<td>29</td>
<td>Sydney Exhouse</td>
<td>D</td>
</tr>
<tr>
<td>12</td>
<td>Prospect Res</td>
<td>J</td>
<td>30</td>
<td>Sydney Inhouse</td>
<td>D</td>
</tr>
<tr>
<td>13</td>
<td>Pheasants Nest Weir</td>
<td>G</td>
<td>31</td>
<td>South Illawarra Inhouse</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>Pacific Ocean</td>
<td>W</td>
<td>32</td>
<td>South Illawarra Exhouse</td>
<td>D</td>
</tr>
<tr>
<td>15</td>
<td>Warragamba inflow</td>
<td>S</td>
<td>33</td>
<td>South Illawarra Commercial</td>
<td>D</td>
</tr>
<tr>
<td>16</td>
<td>Wingecarribee inflow</td>
<td>S</td>
<td>34</td>
<td>Desalination plant</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>L.Yarrunga Dummy</td>
<td>G</td>
<td>35</td>
<td>Desalination junction</td>
<td>J</td>
</tr>
<tr>
<td>18</td>
<td>Burrier Pumping station</td>
<td>P</td>
<td>36</td>
<td>Fitzroy Fall splitter</td>
<td>J</td>
</tr>
</tbody>
</table>
Table 6-4 Description of nodes appearing in the schematic shown in Figure 6.8
(From Kuczera 1997)

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Consumptive demand</td>
<td>Node can represent an urban, industrial or irrigation zone having a consumptive demand. No water is returned to the water supply network. A storage-based restriction policy can be applied to reduce demand.</td>
</tr>
<tr>
<td>R</td>
<td>Reservoir</td>
<td>Node represents a reservoir whose storage volume can be regulated. It may represent either surface or sub-surface storage. Surface storages located on a stream can receive unregulated streamflow as well as regulated inflows. Surface storages located off-stream can receive inflow from conduits and rainfall. Sub-surface storages can be recharged by stream channels, rainfall and conduits.</td>
</tr>
<tr>
<td>G</td>
<td>Gravity diversion</td>
<td>Node represents a weir (with negligible storage volume) which diverts water from a stream into a conduit. Flow energy comes from gravity. Hence operating cost is minimal. Unit costs on all conduits leaving the diversion should be zero. Unregulated streamflow can flow to the node. More than one stream channel can flow to a diversion.</td>
</tr>
<tr>
<td>P</td>
<td>Pump diversion</td>
<td>This node is the same as a gravity diversion except that a pump provides the energy to divert water. Pump operating cost is modelled by setting appropriate unit costs on conduits leaving the diversion. Unregulated streamflow can flow to the node.</td>
</tr>
<tr>
<td>J</td>
<td>Conduit junction</td>
<td>Node represents a confluence of conduits.</td>
</tr>
<tr>
<td>S</td>
<td>Stream junction</td>
<td>Node represents a confluence of streams. Unregulated streamflow can flow to the node.</td>
</tr>
<tr>
<td>W</td>
<td>Waste node</td>
<td>Stream channels must form a tree-like network draining into a single node called a waste node. Only one such node is allowed in a network.</td>
</tr>
</tbody>
</table>
Table 6-5 Description of arcs used in WATHNET (From Kuczera 1997)

<table>
<thead>
<tr>
<th>Type</th>
<th>Line type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>Dashed</td>
<td>Arrow points in the downstream direction. The stream channel network is assumed to have a tree-like structure and must drain into a single waste node; Instream flow requirements can be defined in five ways: i) fixed; ii) seasonal; iii) dependent on the demand at a nominated demand node; iv) dependent on the flow in a nominated arc; or v) dependent on the volume in a nominated reservoir. Maximum flow capacities are usually set to an arbitrarily large value to convey flood volumes.</td>
</tr>
<tr>
<td>Conduit</td>
<td>Solid</td>
<td>May be a pipe or a channel. Arrow points in flow direction. It is permissible to have multiple conduits flowing in the same direction between any two nodes. Two-way conduits can be modelled as one-way arcs flowing in opposite directions. Maximum flow capacities can be defined in more than five ways: i) fixed; ii) seasonal; iii) dependent on the demand at a nominated demand node; iv) dependent on the flow in a nominated arc; or v) dependent on the volume in a nominated reservoir.</td>
</tr>
</tbody>
</table>
Figure 6.11 Illustration of Warragamba pump mark in the WATHNET

2. Desalination is an emerging water supply option and will be explored in this case study. In Figure 6.10, node 34 represents the desalination plant. The arc from node 34 to 35 has a capacity equal to the capacity of the desalination plant. Node 35 allows desalinated water to be transferred to the Sydney and South Illawarra zones. If the Warragamba storage level falls below the desalination trigger level, construction of the desalination plant commences. It is assumed that the plant will be commissioned 12 months after the decision to build.
3. For purpose of demand simulation, the Sydney region is subdivided into two zones: Sydney Metropolitan and South Illawarra zones. Each of these zones has three demand categories; indoor domestic, outdoor domestic and commercial. Nodes 28 to 33 in Figure 6.10 represent these zones and categories. Further details are provided in a subsequent section.

6.3.3.1 Demand Generation for Sydney Headworks System

This section discusses the second input data used by the WATHNET model, namely the demand data for the inhouse, exhouse and commercial nodes served by the Sydney headworks system.

6.3.3.2 Characteristics of Sydney Water Use

Urban water demands are frequently broken up into a number of groups, such as water for industrial, commercial, public and residential use. Table 6-6 shows the breakdown of demand in the Sydney area corresponding to an average water use of 460 litres/capita/day. Residential usage forms the largest component of demand - accounting for more than half of the total water consumption, with industrial and commercial consumption being the other significant categories of water use.

Typically, residential water is disaggregated into indoor and outdoor consumption. It has the most significant impact on water demand and patterns of demand. Over the period of record for all industrialised cities, residential water use has continued to rise on a per capita basis. Indoor residential use remains reasonably constant throughout the year with little climate effect. Table 6-7 shows average garden water use for major Australian cities. It suggests that garden water use account for 30% to over 60% of annual residential water use in the major cities, with the actual amount varying according to type and size of garden and local climatic conditions.
Table 6-6 Breakdown of Demand in Sydney (litres/capita/day)  
(From Eberhardt et al. 1983)

<table>
<thead>
<tr>
<th>City</th>
<th>Residential</th>
<th>Commercial</th>
<th>Industrial</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sydney</td>
<td>239</td>
<td>79</td>
<td>85</td>
<td>55</td>
<td>460</td>
</tr>
</tbody>
</table>

Table 6-7 Estimate of garden water use as a percentage of total residential use (Eberhardt et al. 1983)

<table>
<thead>
<tr>
<th>City</th>
<th>Estimate * (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sydney</td>
<td>31</td>
</tr>
<tr>
<td>Melbourne</td>
<td>36</td>
</tr>
<tr>
<td>Adelaide</td>
<td>44</td>
</tr>
<tr>
<td>Perth</td>
<td>63</td>
</tr>
<tr>
<td>Canberra</td>
<td>63</td>
</tr>
</tbody>
</table>

*Estimate based on average daily inhouse usage of 165 litres/capita/day.

6.3.3.3 Disaggregation of Sydney Water Demand

For simplicity, the demand area is divided into two zones in this study: Sydney Metropolitan and South Illawarra. Each demand zone is further classified into domestic use, consisting of indoor and outdoor water consumption, and non-residential (commercial + industrial) use. Therefore, there are three demand categories in each demand zone: indoor water use, outdoor water use and commercial + unaccounted water uses. Table 6-8 shows the six demand categories used in this case study.
Table 6-8 Classification of Sydney water demand

<table>
<thead>
<tr>
<th>Demand zone</th>
<th>Demand name</th>
<th>Demand category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sydney Metropolitan</td>
<td>Demand (1)</td>
<td>Indoor water consumption</td>
</tr>
<tr>
<td></td>
<td>Demand (2)</td>
<td>Outdoor water consumption</td>
</tr>
<tr>
<td></td>
<td>Demand (3)</td>
<td>Commercial + unaccounted-for</td>
</tr>
<tr>
<td>South Illawarra</td>
<td>Demand (4)</td>
<td>Indoor water consumption</td>
</tr>
<tr>
<td></td>
<td>Demand (5)</td>
<td>Outdoor water consumption</td>
</tr>
<tr>
<td></td>
<td>Demand (6)</td>
<td>Commercial + unaccounted-for</td>
</tr>
</tbody>
</table>

6.3.3.4 Per Capita Domestic Demand

No data was available regarding per capita indoor and outdoor water consumption for the Sydney region. Noting that the Sydney region has a similar climate and lifestyle to the Newcastle region, historical domestic water consumption data from the Newcastle region is therefore used to disaggregate Sydney domestic water demand. Table 6-9 shows the monthly per capita average for indoor and outdoor water consumption ranging from 1987 to 1994 in the Newcastle region. As seen in Table 6-9, indoor water consumption remains reasonably constant throughout the year with little climate effect. However, there exist significant variations for the outdoor water use over the year.

Table 6-9 Average monthly per capita indoor and outdoor water consumption (kL/capita/month) (Coombes 2002)

<table>
<thead>
<tr>
<th>Month</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>InHouse</td>
<td>3.98</td>
<td>3.89</td>
<td>4.15</td>
<td>4.10</td>
<td>4.31</td>
<td>4.05</td>
<td>4.22</td>
<td>4.20</td>
<td>4.01</td>
<td>4.23</td>
<td>4.17</td>
<td>4.29</td>
</tr>
<tr>
<td>ExHouse</td>
<td>2.17</td>
<td>1.56</td>
<td>1.29</td>
<td>0.98</td>
<td>0.60</td>
<td>0.55</td>
<td>0.65</td>
<td>1.01</td>
<td>1.57</td>
<td>2.10</td>
<td>2.01</td>
<td>2.12</td>
</tr>
</tbody>
</table>
6.3.3.5 Regression Analysis

There are many factors affecting water consumption, such as climate, standard of living and water tariff. Because the focus of the case study is to demonstrate the optimisation approach, this study simplifies the analysis and considers only the effect of climate factors on outdoor water consumption. The relationship is established for synthetically generating future water demand.

The monthly rainfall data and the number of rain days for each corresponding month are shown in Appendix 3 and 4. A monthly regression analysis between per capita outdoor water consumption and rainfall depth or number of rain days was performed. The monthly regression model was calibrated using data from January 1987 to January 1994.

The regression equation is assumed to be in the following form:

\[ Y(i) = A(i) + B(i) \times X(i) \]  \hspace{1cm} (6.4)

where \( Y(i) \) is the outdoor water consumption in kilolitres/capita for the \( i^{th} \) month, and \( X(i) \) is either monthly rainfall in mm or the number of rain days per month. Table 6-10 shows the coefficients of a linear regression analysis between outdoor water consumption and rain days or rainfall depths obtained using Newcastle data (Coombes 2002). Overall, the correlation R favoured the use of rain days as the explanatory variables. Accordingly, rain days were used to estimate future outdoor water use in the following section.
Table 6-10 Regression analysis summary

<table>
<thead>
<tr>
<th>Month</th>
<th>Outdoor water use (kL/capita)</th>
<th>Outdoor water use (kL/capita)</th>
<th>Vs Rain days</th>
<th>Vs Rainfall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A(i)</td>
<td>B(i)</td>
<td>R</td>
<td>A(i)</td>
</tr>
<tr>
<td>January</td>
<td>5.447</td>
<td>-0.264</td>
<td>0.632</td>
<td>3.871</td>
</tr>
<tr>
<td>February</td>
<td>3.294</td>
<td>-0.153</td>
<td>0.762</td>
<td>2.108</td>
</tr>
<tr>
<td>March</td>
<td>3.174</td>
<td>-0.122</td>
<td>0.950</td>
<td>1.931</td>
</tr>
<tr>
<td>April</td>
<td>2.224</td>
<td>-0.118</td>
<td>0.886</td>
<td>1.323</td>
</tr>
<tr>
<td>May</td>
<td>0.644</td>
<td>0.000</td>
<td>0.047</td>
<td>0.313</td>
</tr>
<tr>
<td>June</td>
<td>0.861</td>
<td>-0.027</td>
<td>0.573</td>
<td>0.658</td>
</tr>
<tr>
<td>July</td>
<td>0.864</td>
<td>-0.027</td>
<td>0.718</td>
<td>0.798</td>
</tr>
<tr>
<td>August</td>
<td>1.491</td>
<td>-0.077</td>
<td>0.864</td>
<td>1.219</td>
</tr>
<tr>
<td>September</td>
<td>2.424</td>
<td>-0.092</td>
<td>0.558</td>
<td>2.232</td>
</tr>
<tr>
<td>October</td>
<td>3.248</td>
<td>-0.099</td>
<td>0.761</td>
<td>2.936</td>
</tr>
<tr>
<td>November</td>
<td>3.512</td>
<td>-0.132</td>
<td>0.752</td>
<td>1.916</td>
</tr>
<tr>
<td>December</td>
<td>3.748</td>
<td>-0.148</td>
<td>0.631</td>
<td>2.958</td>
</tr>
</tbody>
</table>

6.3.3.6 Population Projection

Population is considered to be the key explanatory variable to forecast future water demand. Table 6-11 presents the population projections for the years 2000, 2010, 2020, 2030, 2040 respectively (ABS 1998). This population projection is used in this study to determine the future water demands.

Table 6-11 Population used for the demand model

<table>
<thead>
<tr>
<th>Year</th>
<th>2000</th>
<th>2010</th>
<th>2020</th>
<th>2030</th>
<th>2040</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population</td>
<td>4056300</td>
<td>4405700</td>
<td>4714400</td>
<td>4975300</td>
<td>5050200</td>
</tr>
</tbody>
</table>
6.3.3.7 Estimating Future Monthly Demand

This section describes in detail how to estimate future water demand for each category. The following assumptions are made in order to project Sydney’s future water demand:

1. Domestic water usage forms the largest component of Sydney’s demand, accounting for 55% of the total demand, which can be expressed as:

\[
\frac{\text{Domestic demand}}{\text{Total demand}} = 55\%
\]

where the total demand is the sum of residential, and commercial water use plus unaccounted-for water.

2. Unaccounted-for water is assumed 15% of total water demand. The unaccounted-for water is defined as the difference between the amount delivered and the amount consumed. Therefore, commercial (industrial) water consumption accounts for 30% of the total water demand. Unaccounted--for water will be lumped with commercial water use to simplify the headworks simulation.

3. It is assumed that 90% of the total population resides in the Sydney Metropolitan zone, and only 10% resides in the South Illawarra zone.

4. It is assumed that the monthly inhouse water consumption is constant for each month of the simulation period.

5. Outdoor water consumption during the simulation year is only affected by the number of rain days and the time of year.

Monthly demand data are required for each of the six demand categories to simulate the Sydney water supply headworks system. The simulation consists of 1000 replicates \((i = 1, \ldots, 1000)\) with each replicate ranging from year 1995 to 2045 \((t = 1995, \ldots, 2045)\) at a monthly resolution \((k = 1, \ldots, 12)\). The following steps were implemented to determine water demand at each node:

(i) Generate the monthly average indoor per capita water consumption \((Inwater(k), k = 1, \ldots, 12)\) using the data in Table 6-9.

(ii) Generate concurrently with streamflow the number of rain days at Prospect reservoir denoted as \(Rain^i_t(k)\)
(iii) Compute the total monthly demand (Megalitres or ML) for the following three categories:

**Indoor residential demand:**

\[ \text{InHouse}^i_k = \text{inWater}(k) \times \text{Population}(t, k) \]  \(6.6\)

where Population\((t, k)\) is the population for month \(k\) and year \(t\).

**Outdoor residential demand:**

\[ \text{ExHouse}^i_k = [A(k) + B(k) \times \text{Rain}^i_k] \times \text{Population}(t, k) \]  \(6.7\)

**Commercial demand and unaccounted-for water:**

\[ \text{Commercial}^i_k = [\text{InHouse}^i_k + \text{ExHouse}^i_k] \times 0.45/0.55 \]  \(6.8\)

(iv) With reference to Table 6-8, estimate future water demand for each demand category (indoor, outdoor, and commercial demand) and each simulation year on a monthly basis:

\[ \text{demand}^i(1) = \text{inHouse}^i_k \times 0.90 \]  \(6.9\)

\[ \text{demand}^i(2) = \text{exHouse}^i_k \times 0.90 \]  \(6.10\)

\[ \text{demand}^i(3) = \text{commercial}^i_k \times 0.90 \]  \(6.11\)

\[ \text{demand}^i(4) = \text{inHouse}^i_k \times 0.10 \]  \(6.12\)

\[ \text{demand}^i(5) = \text{exHouse}^i_k \times 0.10 \]  \(6.13\)
The above procedure was implemented using a Fortran 90 program to produce a data file in a form that could be read by the WATHNET model.

**6.3.4 Scope For Optimising Existing Operating Rules**

The drawdown curves represent a significant portion of operating rules as they define which reservoir should be releasing water as a function of the current storage volumes in other reservoirs. As described in Section 6.3.2.1, the drawdown curves can be readily incorporated into the WATHNET model and its arrangement in WATHNET is fairly flexible. Furthermore, the NetLP in the WATHNET optimises water assignments within the network and in doing so optimises, to some extent, the distribution of water within the system. From this point of view, the current drawdown curves do not warrant further optimisation in this study.

Of the three major operating rules applied to the Sydney headworks system, system drought security (as measured by the probability of experiencing restrictions at each of the four levels) is most affected by the selection of restriction trigger levels and the Shoalhaven pump mark.

The appropriate selection of restriction levels is particularly important. As the trigger level for level 1 restrictions is lowered, the chance of entering restrictions is reduced at the expense of a greater risk of experiencing level 4 restrictions and running out of water during a severe drought.

The selection of an appropriate pump mark is important as well. As Sydney’s water demand increases, the pump mark has to be raised to maintain system reliability; this in turn will progressively increase the cost of pumping. Studies done by Sydney Water Board (1991) indicated that the range of pump marks that may be considered practical is between 65% and 95%. Selection of a high pump mark is likely to result in a greater chance of reservoir spillage and hence an expensive wastage of pumped Shoalhaven water. Furthermore, keeping Warragamba reservoir full by transferring from the Shoalhaven system reduces the flood mitigation effect.

\[
demand \hat{\gamma}(6) = \text{commercial} \hat{\gamma}(k) \times 0.10
\]  

(6.14)
of the dam. Selection of a low pump mark may prove to be disastrous for drought security. By the time the system draws down to a low pump mark, unregulated river flow in the Shoalhaven may be very low – there may be no water to transfer to Warragamba! The selection of an appropriate pump mark necessitates a careful trade-off between drought security and pumping costs.

6.4 SUMMARY

This chapter introduced the Sydney headworks case study which will be used to assess the utility of probabilistic search methods to optimise complex headworks system. The chapter first described the WATHNET simulation model to be used in this study. It then described the input data required for the WATHNET simulation model: the streamflow and the demand data. The generated streamflow and the demand data are stored in output files for use by the WATHNET. Finally it described the application of the WATHNET model to Sydney headworks system and explored opportunities to further optimise its operating policy. Particular attention was directed to the selection of restriction trigger levels and the Shoalhaven pump mark, which are considered to have most effect on drought security. The subsequent chapters will consider the following questions:

- What are the most appropriate levels at which restrictions should commence for different population scenarios?

- What are the optimal pump marks (for Warragamba and Avon reservoirs) in the sense of avoiding unnecessary spill of pumped water yet ensuring that severe drought is survived without catastrophic water shortages?

- What is the value of desalination as an emergence source of water during severe drought?

These questions will be served as the basis to formulate the objective function for the GA optimisation. The next step will be the formulation of the objective function, which is the focus of the next chapter.
CHAPTER 7

CASE STUDY:
FORMULATION OF THE
OBJECTIVE FUNCTION
7.1 INTRODUCTION

As noted in the Chapter 2, an optimisation model requires specification of an objective function, decision variables and constraints on the solution space. The objective of this chapter is to formulate the optimisation problem in the context of the Sydney headworks case study.

Central to the optimisation problem is the specification of the objective function. In the Sydney headworks case study, it was decided to search for decisions that can minimise the total expected cost of operating the headworks system. In Chapter 6, it was concluded that system security was significantly affected by restriction levels and pump marks and hence these operating variables should be subject to optimisation. In addition, the option to provide desalination in the event of a severe drought was introduced. Whereas the evaluation of pumping and desalination costs is quite straightforward, the evaluation of the economic cost of imposing water restrictions presents more of a challenge. Dandy (1989, 1992) and Kuczera and Ng (1994) have argued that the theory of consumer willingness-to-pay for water can be used to evaluate the economic impact of restrictions on outdoor domestic water consumption. When restrictions are imposed, consumers suffer a loss of utility and the water authority may suffer a loss of revenue. Therefore, in this study, the cost of imposing water restrictions is evaluated using consumer surplus theory.

This chapter first reviews the theory of consumer surplus for assessing the economic costs due to outdoor water restrictions. It then describes in detail the decision variables that are to be optimised and finally presents the formulation of the objective function.

7.2 ECONOMIC EVALUATION OF WATER RESTRICTION LOSSES

7.2.1 Theory of Consumer Surplus

Figure 7.1 illustrates a demand curve for a household. Assuming a pay-for-use pricing policy applies, the demand curve expresses the relationship between the price charged for water $p ($/kL) and the resultant household demand $q$ (kL/month) (Brent
An important property of the demand curve is given by the price elasticity $\epsilon$ defined as:

$$\epsilon = \frac{\Delta q}{q} \div \frac{\Delta p}{p}$$

(7.1)

where $\Delta q$ is the change in demand arising from a change in price $\Delta p$.

If the demand curve is continuous and the quantity $q$ is differentiable with respect to price $p$, $\epsilon$ can be expressed for small changes in price as:

$$\epsilon = \frac{dq}{q} \div \frac{dp}{p} = \frac{dq}{dp} \cdot \frac{p}{q}$$

(7.2)

It is clear that the price elasticity of demand is negative for a downward sloping demand curve. It provides a convenient way of describing the response in demand to changes in price. The greater is the absolute value of $\epsilon$, the more responsive is the demand.

Elasticity is also closely related to the financial impact of a price change on the utility: If $-\infty < \epsilon < -1$, then a decrease in price will increase total revenue from commodity charges (price times quantity). If $\epsilon = -1$, total revenue will not change with price changes. If $-1 < \epsilon < 0$, a decrease in price will decrease total revenue.

Figure 7.1 A demand curve for a typical household
Elasticity generally varies amongst water users who have different consumption volumes, different incomes, different property sizes, different businesses, and other individual or sub-group characteristics, as well as by season, climate and other factors.

Numerous empirical studies have been made of price elasticity for domestic water use. Howe and Linaweaver (1967) found an average annual elasticity of –0.40 for 21 metered areas, which means that an increase in price of 100% would cause a 40% drop in demand. Hank (1978) reported price elasticities ranging between -0.1 to -1.0. A review by Danielson (1979) surveyed demand studies with price elasticity ranging between -0.3 and -0.7. Cassuto and Ryan (1979) reviewed 13 other published studies and reported price elasticities ranging from -0.02 to -0.75 with a median of about -0.5. Gallagher and Robinson (1977) reported that the elasticity could be expected to lie between –0.2 and –0.5. Generally, it seems that indoor water uses may be expected to be more price inelastic, while outdoor water uses for swimming pools, gardening and the like exhibit great price elasticity.

Montgomery Watson and SKM (1995) reported elasticities for served properties in the Hunter region ranging from -0.7 to -0.05 with -0.20 being the average; and the elasticity in the Sydney area was estimated to be -0.05 for indoor use and -0.3 for outdoor residential use.

### 7.2.2 Consumer Surplus and Water Restrictions

The economic benefit from domestic water use is defined by the consumer surplus, which is the difference between the maximum amount that a consumer would be willing to pay for water consumption and the amount that is actually paid. During a drought period when restrictions are imposed, the domestic consumer experiences an economic loss in the sense that he/she forgoes consumption which, under normal circumstances, he/she was willing to pay for. Dandy (1989, 1992) provided a detailed analysis of an economic approach evaluating water restriction losses and it will be briefly reviewed below.
Suppose a water authority supplies water at a price of $p_1$ ($/kL)$ with no limit on the allowable consumption by an individual household. A demand curve for a particular household is illustrated in Figure 7.2. The supply curve for the household is a horizontal line through $p_1$. The total benefit to the household of consuming this water is given by its total willingness to pay for the water, which equals the area under the demand curve up to $q_1$ (area $ocaq_1$). Some of the benefit is transferred to the water authority in payment for supplying the water (area $op_1aq_1$). The balance is the consumer surplus (area $cap_1$) and equals the net benefit to the customer. The total revenue to the water authority will be partly offset by the economic cost of supplying the water to all consumers. The total net benefit to the community as a whole (consumers plus water authority) is the sum of the consumer surplus for all consumers plus the revenue of the water authority minus its supply costs.

In case of drought, if restrictions have to be imposed by increasing the water price from $p_1$ to $p_2$, the net loss of benefits to the consumer and supply authority is the sum of the loss of consumer surplus (sum of area 2 and 3 in Figure 7.2) plus the loss of

![Figure 7.2 Household demand-price relationship showing loss of revenue and consumer surplus due to imposition of restrictions](image-url)
revenue (area 1 minus area 2 in Figure 7.2) (Dandy 1989).

However, there are very few water authorities using pricing to ration water use during a drought. The reluctance on the part of water managers to use a price mechanism for rationing probably relates to questions of equity, political constraints and uncertainty about the effectiveness of such a mechanism. In practice, water restrictions are used by many water authorities as a method for rationing water during a drought. Mandatory water restrictions may take a variety of forms such as: restricting the time of use of garden sprinklers; banning of sprinklers entirely and permitting only hand-held hoses for garden watering; banning all gardening watering, and limiting the quantity of water which a household may use in a specific time period (Dandy 1992).

Dandy (1992) developed a method to evaluate the economic cost of stringent time restrictions on outdoor water use. A comparison was made for estimating these costs with the economic costs of an equivalent price increase. Particular examples were developed for the case of households with linear demand functions and households with demand functions of constant elasticity. It was shown that the economic cost of stringent time restrictions was greater than that for an equivalent price increase. The difference was due to variations in the demand curves between households and variations in the responses of individual households to water restrictions. The economic cost of mandatory water restrictions is equal to that for an equivalent price increase in the special case where no such household variations exist.

For simplicity, this case study considers the economic cost of restrictions only for the special case. Referring to Figure 7.2 the loss of consumer surplus is given by area 3 and the loss of revenue by area 1 yielding a total loss defined by area $baq_1q_2$.

### 7.2.3 Estimation of Economic Loss Due to Water Restrictions

Following Dandy (1989, 1992), the total loss of benefits due to restrictions on outdoor water use can be readily derived under the following assumptions:

1. All the households have the same price elasticity of demand for outdoor use.
2. The price elasticity for outdoor use is constant within the range considered.

3. All households reduce their outdoor consumption in the same proportion in response to water restrictions.

Referring to Figure 7.2, at the current price $p_1$ ($/kL), the household has an outdoor consumption of $q_1$ (kL/month). If restrictions are imposed, consumers are forced to reduce their consumption from $q_1$ to $q_2$ with a shadow price of $p_2$ ($/kL). As shown in Figure 7.2, the change in willingness to pay (area ba$q_1q_2$) is the same as for an equivalent price increase.

Define the fractional reduction in water consumption of a particular household in response to water restrictions as:

\[ R = \frac{(q_1 - q_2)}{q_1} = 1 - \frac{q_2}{q_1} \quad (7.3) \]

Then under the assumption of constant price elasticity, the following demand equation can be obtained from the equation (7.2).

\[ p = p_1 \left( \frac{q}{q_1} \right)^{1/\varepsilon} \]
and \[ q = q_1 \left( \frac{p}{p_1} \right)^{\varepsilon} \quad (7.4) \]

Integrating from $q_2$ to $q_1$ the total loss of economic benefits due to water restrictions (area 1 + area 3 in Figure 7.2) can be estimated by:

\[ T = \int_{q_2}^{q_1} p dq \]
\[ = \frac{\varepsilon}{1 + \varepsilon} p_1 q_1 \left[ 1 - \left(1 - R \right)^{1+\varepsilon/\varepsilon} \right] \quad (7.5) \]

\[ \text{for } \varepsilon \neq -1 \]

\[ T = - p_1 q_1 \ln(1 - R) \quad \text{for } \varepsilon = -1 \quad (7.6) \]
Dandy (1992) has shown that the above analysis underestimates the actual loss due to demand shortfalls. The above equation, therefore, provides a lower bound on the economic loss due to restrictions when implemented in the form of time or quantity restrictions.

7.2.4 Modified Cost of Water Restriction Policies

Equation (7.5) can be used to estimate the economic loss due to water restrictions. However, when 100% water restrictions are applied (namely \( q_2 = 0 \) and hence \( R = 1 \)) equation (7.5) will produce an infinite value of \( T \). Clearly there is a need to modify the equation to avoid infinite \( T \) as \( R \) approaches to 1, which is the issue of this section.

The demand-price relationship is modified to avoid this singularity. A critical value for outdoor water consumption \( q_x \) in Figure 7.3) is introduced below which a constant price \( p_{\text{max}} \) is imposed. The price \( p_{\text{max}} \) could be interpreted as the cost of an alternative source of water supply such as water imported using tankers or provided by desalination. Therefore, the price demand relationship can be redefined as:

\[
p = \begin{cases} 
\frac{p_1 q}{q_i}^{1/\varepsilon} & q \geq q_x \\
p_{\text{max}} & q < q_x 
\end{cases} \tag{7.7}
\]

The expected cost of water restrictions is given in the following equations and illustrated in Figures 7.3 and 7.4.

1. If \( q_2 \geq q_x \), Equation (7.5) can be used to estimate the total loss due to water restrictions.

2. If \( q_2 < q_x \), then

\[
T = \int_{q_2}^{q_1} pdq = \int_{q_1}^{q_1} p_1 (q / q_1)^{1/\varepsilon} dq + \int_{q_2}^{q_1} p_{\text{max}} dq \\
= (\varepsilon'(1 + \varepsilon)p_1 q_i \{1 - (q_x / q_i)^{(1+\varepsilon)/\varepsilon}\} + (q_x - q_2) p_{\text{max}} \quad \text{for } \varepsilon \neq -1 \tag{7.8}
\]
\[ T = -p_1q_1 \ln \left( \frac{q_1}{q_x} \right) + \int_{q_1}^{q_x} p_{\text{max}} dq \]

\[ = -p_1q_1 \ln \left( \frac{p_{\text{max}}}{p_1} \right)^{\varepsilon} + (q_x - q_2)p_{\text{max}} \quad \text{for } \varepsilon = -1 \quad (7.9) \]

where, from equation (7.4), \( q_x \) can be expressed as:

\[ q_x = q_1 \left( \frac{p_{\text{max}}}{p_1} \right)^{\varepsilon} \quad (7.10) \]

and, from equation (7.3), \( q_2 \) can be derived from \( q_1 \) and \( R \) using:

\[ q_2 = q_1 (1-R), \quad (7.11) \]

Equations (7.8) to (7.11) are used to estimate the cost of restricting outdoor water use in this study. The following values were considered indicative and adopted for the Sydney case study: \( p_1 = $0.6/kL, \varepsilon = -0.25, p_{\text{max}} = $1.2/kL. \)

![Figure 7.3 Cost of restrictions when \( q_2 \geq q_x \)](image)
7.3 OBJECTIVE FUNCTION FORMULATION

7.3.1 Decision Variables

In this case, study, the optimisation problem formulated for the Sydney headworks system has been somewhat simplified in keeping with the objective of demonstrating the use of parallel GA. As described in Chapter 6, of the three existing operating rules applied to the Sydney headworks system, the reservoir drawdown curves are readily incorporated into the WATHNET model and do not warrant further optimisation. System drought security is most affected by the selection of restriction trigger levels and the Shoalhaven pump marks. Therefore, the selection of appropriate restriction levels and appropriate pump marks is particularly important. In addition, the case study also considers an emergency source to alleviate water shortage in the event of an extreme drought, namely a desalination plant. The decisions involving the desalination plant include the option of when to build it and deciding on its operating capacity. Therefore, the decision variables that require optimisation in this case study are:
- Decision (1) to Decision (4) represent the four storage levels which trigger different levels of outdoor water restrictions.

- Decision (5) and Decision (6) represent the pump marks below which pumping from Shoalhaven will commence to supply two demand zones, the Sydney Metropolitan area and the South Illawarra area.

- Decision (7) and Decision (8) define the desalination option. Decision (7) is the trigger storage volume in Warragamba below which construction of the desalination plant commences. Decision (8) is the capacity of desalination plant. Once the decision to commence construction of the desalination plant is made, it is assumed it will take 12 months before it is commissioned.

### 7.3.2 Objective Function

The objective of the optimisation problem is to minimise the present worth of the expected operating cost to meet demand corresponding to a particular population scenario. For a given set of decision variables proposed by the GA, the WATHNET simulation model performs Monte Carlo analysis to evaluate the objective function according to the following equations:

\[
F = \text{obj}F + C_F
\]

\[
\text{obj}F = \frac{1}{\text{rep}} \times \sum_{i=1}^{\text{rep}} \frac{1}{\text{simyrs}} \sum_{t=1}^{\text{simyrs}} P_w(t) \times [C_R(i,t) + C_F(i,t) + C_D(i,t) + C_C(i,t)]
\]

where \( \text{obj}F \) is the present worth of the expected annual operating cost and \( C_F \) is the penalty cost of violating system reliability constraints. The subscript \( t \) represents the simulation year, \( \text{simyrs} \) is the maximum simulation years, namely 50 years; \( i \) is the replicate number; \( \text{rep} \) is the maximum number of replicates for Monte Carlo simulation. \( P_w(t) \) is the present worth defined as \((1 + r)^t\) with \( r \) being the discount rate.

It is important to appreciate a few subtle points about the case study. The optimisation seeks to find near-optimal decisions for a particular population scenario,
which coincides with a particular point in time. Therefore, the simulation model needs to describe the steady-state behaviour of the system corresponding to the population scenario. It, therefore, is appropriate to disable the present worth term in equation (7.13) by setting the discount rate to zero. In that way, the objective function yields the expected annual operating cost for a particular point in time. Ideally, the simulation should run for a very long time (i.e. simyrs is large) in order that steady-state performance be obtained. Because of the way the WATHNET model organises computer memory, this is not practical. Accordingly, multiple replicates were used with simyrs set to a sufficiently large value (50 years) to minimise the effect of the initial conditions and hence approximate steady-state behaviour of the headworks system. Because the primary objective is to demonstrate the capability of parallel GA this approximation is considered acceptable.

The objective function has five components. The first component $C_R$ is the cost of imposing restrictions on outdoor water use. The cost is incurred only if Warragamba storage falls below the restriction trigger level. The second component $C_P$ is the cost of pumping from the Shoalhaven system, which includes the cost of pumping for water supply to the Sydney Metropolitan and South Illawarra areas. The third component $C_D$ has two terms, $C_{D1}$ representing the cost of building a desalination plant and $C_{D2}$ the cost of producing water once the plant is commissioned. The fourth component $C_C$ is the cost of rationing commercial and/or residential indoor water use. The last component $C_F$ is the penalty for not meeting a minimum system reliability of 90%. The five components all have units of dollars. It is stressed that the WATHNET is run using monthly time interval. Each of the five components therefore represents the sum of monthly costs for each simulation year. The mathematical representation of each component is described in the following subsections.

7.3.2.1 $C_R$ - Penalty of Imposing Water Restrictions on Outdoor Water Use

Section 7.2 described how the theory of consumer surplus can be used to calculate the economic cost due to water restrictions. Defining Level($j$), $j = 1,\ldots, 4$ as the
percentage of Warragamba storage at which the jth level of restrictions is triggered, equations (7.8) to (7.11) can be used to calculate $C_R$.

Because the trigger levels must form a decreasing sequence (i.e. $\text{Level}(j+1) < \text{Level}(j)$) some care is required in specifying the decision variables. In the GA the decision space is a hypercube spanning from 0 to 1. No constraints are imposed on decisions within this hypercube. To avoid generating infeasible decisions, it is essential that a decreasing sequence for trigger levels be generated using the following mapping:

$$\text{Level}(j) = \text{Level}(j+1) + [100-\text{Level}(j+1)] \times \text{Decision}(j), \ j = 4, \ldots, 1 \quad (7.14)$$

where $\text{Level}(5)$ is an additional level set to 10% to ensure that the minimum trigger level never falls below 10% of Warragamba storage. This constrains the search space by eliminating solutions that a priori are judged to be unacceptable.

Figure 7.5 illustrates how equation (7.14) preserves the trigger level ordering. It is noted that $\text{Decision}(j), \ j=1, \ldots, 4$ corresponds to the restriction trigger levels in GA decision space. Using equation (7.14) the fourth trigger level $\text{Level}(4)$ is determined first, which is the sum of the minimum level, $\text{Level}(5)$, and the fraction of $(100-\text{Level}(5))$ as given by $\text{Decision}(4)$. The third trigger level $\text{Level}(3)$ is then dependent on the fourth level and the fraction of $(100-\text{Level}(4))$, and so on.

7.3.2.2 $C_P$ - Cost of Pumping From The Shoalhaven System

As described in Chapter 6, two pump marks are required in this case study. The first pump mark is the Warragamba storage volume below which water from the Shoalhaven system is transferred to Warragamba reservoir for distribution to the Sydney Metropolitan demand zone. The second pump mark is the Avon storage volume below which water from the Shoalhaven system is transferred to Avon reservoir for distribution to the South Illawarra demand area.
To avoid generating infeasible decisions, the following mapping is used to obtain the pump marks:

\[
pumpMark(1) = (25.0 + 74.0 \times \text{Decision}(5)) \tag{7.15}
\]

\[
pumpMark(2) = (25.0 + 74.0 \times \text{Decision}(6)) \tag{7.16}
\]

This mapping ensures the pump marks lie between 25% and 99%. A pump mark below 25% is judged infeasible because the Shoalhaven system would almost certainly be drought affected and have negligible streamflow.

Figure 7.5 Mapping between four restriction trigger levels and decision variables
The cost of pumping from the Shoalhaven is

\[ C_p = 20 \times \sum_{i=1}^{12} [P_w(i,t,k) + P_A(i,t,k)] \]  \hspace{1cm} (7.17)

where \( i \) is the replicate number; \( t \) is represents the simulation year, and \( k \) is the month interval of simulation year \( t \). \( P_w(i,t,k) \) is the volume of water pumped (ML) from the Shoalhaven system to supply the Sydney Metropolitan demand zone, and \( P_A(i,t,k) \) is the volume of water pumped (ML) for the South Illawarra demand zone.

By assuming a pumping efficiency of 0.75, an energy cost of 0.9 cents/kW-hr, lifting 1 ML water 600 meters high to transfer out of the Shoalhaven basin would cost $20.

### 7.3.2.3 C\(_D\) - Cost of Using Desalination Option

Desalination is used as an emergency water supply option in this case study. Desalination is generally regarded as an expensive and high energy-consumption option. However, the economics of desalination have become more competitive and attractive. Linstrum et al. (2000) provided an overview of the use of desalination in Western Australia. Various desalination processes were examined and capital and operating costs were reported. Indicative costs for several desalination processes for plant sizes between 10 ML/day and 20 ML/day were presented. Linstrum et al.’s indicative costs were adopted for this case study as follows:

1. The cost of commissioning a new desalination plant is $2 million per ML installed capacity per day with a limit on installed capacity of 500 ML/day.

2. The total operating cost of producing desalinated water from the plant is $2000/ML.

The desalination option is implemented as follows. If during a replicate, the Warragamba storage level falls below the desalination trigger level, construction of the desalination plant commences. After 12 months, the plant will be ready to provide water.
Decision (7) is used to obtain the desalination trigger level $trigS$, which is required to be less than the fourth restriction level in this case study. The trigger level expressed as a percentage of Warragamba reservoir capacity is:

$$trigS = (Decision(4) \times 100 + Level(5)) \times Decision(7)$$  \hspace{1cm} (7.18)

The total cost of desalination $CD$ is:

$$CD = C_{D1} + C_{D2}$$  \hspace{1cm} (7.19)

$$C_{D1} = 2000000 \times \frac{Cap}{30.5}$$  \hspace{1cm} (7.20)

As seen in equation (7.19), the cost of implementing the desalination option has two components: $C_{D1}$ is the cost of building the desalination plant, and $C_{D2}$ is the cost of water produced from the desalination plant to the demand area:

$$Cap = 30.5 \times 500 \times Decision(8)$$  \hspace{1cm} (7.21)

$$C_{D2} = 2000 \times \sum_{k=1}^{12} Production(i,t,k)$$  \hspace{1cm} (7.22)

where $Cap$ is the installed desalination capacity (ML/Month) calculated based on Decision (8), and $Production(i,t,k)$ is the actual monthly volume (ML) of water produced by the desalination plant for the $i^{th}$ replicate and the $k^{th}$ month of the $i^{th}$ simulation year.

### 7.3.2.4 $C_C$ - Penalty for rationing commercial and indoor water use

In addition to outdoor water restrictions, restrictions to indoor and commercial water use might also need to be imposed during a severe drought. Restricting commercial water usage is likely to cause significantly bigger economic losses than restricting outdoor water use. Restrictions on commercial usage are likely to lead to shutdown of commercial/industrial activity. Reasons for shutdown range from public health concerns arising from lack of water for sanitary purposes to loss of water in production cycles. The actual estimation of the economic losses represents a major
undertaking which would be peripheral to this thesis. Accordingly a simple yet reasonable approach is adopted. The economic loss will be assumed to be proportional to the degree of restriction with the constant of proportionality being the gross domestic product (GDP) of the Sydney region. According to the Australian Bureau of Statistics (ABS 2000), the Australian GDP in the year 2000 was $600 billion, of which Sydney accounted for 25%. The monthly GDP of the Sydney region is therefore $12.5 billion. Therefore, if commercial use was restricted by R% in a particular month, the penalty would be $12.5\times R/100$ billions.

The estimation of economic loss due to domestic indoor water restrictions presents a difficult problem. Equation (7.8) could be used to estimate the loss using a price elasticity very close to zero. However, the specification of the indoor elasticity is problematic as is the feasibility of finding a large-scale alternative source other than desalination. It is recognised that adequate water for indoor use represents a fundamental assumption of modern urban living. Restriction of such water would represent a political/social crisis. To acknowledge the seriousness of such restrictions the value judgement is made that the penalty for indoor restrictions should be made equal to the economic loss associated with restriction of commercial use.

As discussed in Chapter 6, there are two demand zones in the headworks system, i.e. the Sydney Metropolitan and South Illawarra areas. Of the total population served by the headworks system, about 90% reside in the Sydney Metropolitan area while 10% reside in the South Illawarra area. The penalty for imposing indoor or commercial restrictions is calculated as follows:

For the Sydney Metropolitan area:
\[\text{penalty} = R \times \text{GDP}_{\text{Sydney}} \times 90\% \] (7.23)

For the South Coast area:
\[\text{penalty} = R \times \text{GDP}_{\text{Sydney}} \times 10\% \] (7.24)
where $R$ is the fractional reduction of water available for commercial or domestic indoor consumption and $GDPSydney$ is the monthly GDP for Sydney.

It is important to stress that restrictions are only imposed on outdoor water use in the WATHNET simulation using trigger levels $Level(j)$, $j = 1,\ldots,4$. Restrictions or rationing of indoor or commercial usage only occurs when the headworks system physically runs out of water.

### 7.3.2.5 $C_F$ - Penalty of system failure

The last component in the objective function is the penalty for not meeting the socially mandated minimum system reliability of 90%. This constraint has been imposed to ensure that the GA in pursuit of a minimum economic loss does not produce solutions which are deemed socially and politically unacceptable. The minimum reliability of 90% is a value judgement. If more than 10% of simulation years experience water shortages, the solution is considered unsatisfactory and is therefore penalised. In this study, a multiplicative penalty function has been adopted with the penalty growing as the reliability drops below 90%.

Define $resFrac$ as the fraction of steps experiencing water shortfalls during the Monte Carlo simulation. If $resFrac$ exceeds 10%, the GA solution is deemed unsatisfactory and a penalty $C_F$ is added to the objective function using the following multiplicative penalty:

\[
C_F = \begin{cases} 
    \text{obj}F \times [10.0 \times (resFrac - 0.1) / 0.1] & \text{if } resFrac \geq 0.1 \\
    0 & \text{if } resFrac < 0.1 
\end{cases} 
\]  

(7.25)

Define $resFrac$ as the fraction of steps experiencing water shortfalls during the Monte Carlo simulation. If $resFrac$ exceeds 10%, the GA solution is deemed unsatisfactory and a penalty $C_F$ is added to the objective function using the following multiplicative penalty:

If $resFrac$ equalled 20%, $C_F$ would be 10 times $objF$. This allows the GA to explore decisions that violate the minimum system reliability with the disincentive of a severe penalty.
7.4 SUMMARY

The theory of consumer surplus was introduced to facilitate the estimation of the cost of imposing restrictions on outdoor water use. The eight decision variables requiring to be optimised in the Sydney case study were described. The objective function was then formulated. The objective function was composed of five penalty values, namely penalties for imposing water restrictions on outdoor water use, pumping, desalination, commercial and indoor water use rationing, and violation of minimum system reliability constraints. This completes the formulation of the case study. The next Chapter will present the results.
CHAPTER 8
CASE STUDY: RESULTS AND DISCUSSION
8.1 INTRODUCTION

A GA customised for the headworks optimisation problems was developed in Chapter 4 and its search ability was demonstrated using a simple case study. It was concluded that the GA could effectively identify near-optimal solutions. However, the major problem was the prohibitive computational requirement. Chapter 5 introduced two approaches to reduce the computational turnaround time to obtain near-optimal solutions. Starting with Chapter 6, a full-scale headworks optimisation problem was introduced. The Sydney headworks system was described, the WATHNET simulation model introduced and its application to the headworks system presented. Chapter 7 described the decision variables and formulated the objective function to be optimised by the GA. This Chapter will present the results for optimising important variables affecting the operation of the Sydney headworks system. It is stressed that a major objective of this thesis is to demonstrate the application of probabilistic search methods in conjunction with Monte Carlo simulation and parallel computing. Accordingly, it suffices that the case study only addresses the key aspects of a full-scale headworks optimisation. For this reason, the reader is reminded that the optimisation results to be presented in this Chapter are only indicative.

This Chapter first summarises the parameters that are required to run the GA. It then presents and discusses the results with the objective of providing insight about the ability of parallel GA to optimise a complex headworks system. Two cases are considered with the first one optimising the four restriction trigger levels to explore the applicability of replicate compression, and the second one addressing the full problem involving optimisation of restriction trigger levels, pump marks and desalination options.

8.2 DATA INPUT

This section summarises the key parameters and data used to implement the parallel GA optimisation.
8.2.1 Parallel Genetic Algorithm Parameters

Chapter 4 conducted a feasibility study for the GA. Sensitivity analysis was carried out to determine the optimal GA parameter settings. The GA parameter settings obtained in Chapter 4 will be used as default values for this case study: crossover rate = 0.9; swap rate for indirect swap operation = 0.3; mutation rate = 1/(population size); inversion rate = 0.9; and population size = 80.

8.2.2 WATHNET Simulation Model Input Data

As discussed in Chapter 6, streamflow/climate and demand data files are required for use by the WATHNET simulation model. Streamflow and climate data consisting of 1000 50-year replicates was generated using WATSTRM. Using the generated climate data five demand files were created for the population expected in the years 2000, 2010, 2020, 2030 and 2040. These five demand files define the five population scenarios reported in the results.

8.3 COMPUTATIONAL EFFICIENCY OF PARALLEL COMPUTING

The parallel system used for optimising the Sydney headworks system is an Athlon-based system consisting of 10 nodes interconnected via an Intel 460T 24 port 10/100 switch. The following are the features of each node: (1) Large memory per node (256MB Ram); (2) local disk on each node (10 GB Hard Disk); (3) full Unix on each node (Linux); (4) high performance processors (AMD Athlon 1.2 GHz). The parallel computations were carried out using the cluster of Athlon processors supervised by PVM and located at the Computing Centre of the Department of Civil, Surveying and Environmental Engineering at the University of Newcastle.

One of the objectives of this thesis is to demonstrate the application of parallel computing to reduce the computational turnaround time. This section presents results on the computational efficiency of parallel computing.
8.3.1 Efficiency of Parallel Computing Algorithm

To evaluate the performance of the parallel GA algorithm, it is necessary to define speedup and efficiency.

In the context of parallel computing, speedup is usually determined by calculating the ratio of the elapsed time taken to solve a problem on a single processor to the elapsed time required solving the same problem on a parallel computer with several identical processors (Tsai et al. 1999). Ideally the speedup should approach the number of processors. The speedup can be defined as

\[
S_p(n) = \frac{T_1}{T_p(n)}
\]  

(8.1)

where \( T_1 \) is the elapsed time for a single processor and \( T_p(n) \) is the elapsed time for \( n \) processors running in parallel.

The efficiency can be expressed as

\[
E_p(n) = \frac{T_1}{n \times T_p(n)} = \frac{S_p(n)}{n}
\]  

(8.2)

It provides an indication of the effective utilisation of the \( n \) processors. An efficiency value equal to 1 indicates that the algorithm can run \( n \) times faster when \( n \) processors are used.

To demonstrate the speedup and efficiency of the Athlon cluster the following experiment was performed. 150 objective function evaluations were performed. Each evaluation involved a simulation using 1000 replicates, each 50 years long, for a scenario using the population expected in the year 2000.
The elapsed time, speedup, and efficiency are tabulated in Table 8-1. The table shows that a single objective function evaluation involving 1000 50-year replicates takes about 186 CPU seconds on a single AMD Athlon 1.2 GHz processor. It also reveals that, as expected, elapsed time is reduced and the speedup increases as the number of the processors increases. It can be seen that a speedup of up to 9.93 was obtained when 10 processors were used, indicating that computational time can be nearly ten times faster when ten processors are used. The speedup is therefore considered virtually proportional to the number of processors that are employed. The efficiency was found to be close to 1, regardless of the number of processors. The reason for this is that the communication time between the master and the slave programs is very short compared to the slave computation time required for the objective function evaluation. The only communication involves passing one set of decisions from the master to the slave and passing the objective function value from the slave to the master. In conclusion, the parallel GA algorithm is at least 96% efficient and thus is able to produce very significant reductions in turnaround time almost inversely proportional to the number of processors.

Table 8-1 Performance of parallel computation for 150 objective function evaluations involving 1000 50-year replicates

<table>
<thead>
<tr>
<th>Number of processors</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed time (s)</td>
<td>27837</td>
<td>13987</td>
<td>7277</td>
<td>3618</td>
<td>2803</td>
</tr>
<tr>
<td>Speedup</td>
<td>1</td>
<td>1.99</td>
<td>3.82</td>
<td>7.69</td>
<td>9.93</td>
</tr>
<tr>
<td>Efficiency</td>
<td>1</td>
<td>0.99</td>
<td>0.96</td>
<td>0.96</td>
<td>0.99</td>
</tr>
</tbody>
</table>
8.3.2 Performance of Parallel GA for Optimising the Sydney Headworks System

This section reports the parallel GA performance for optimising the Sydney headworks system. Table 8-2 presents the elapsed time for optimising eight decision variables. The first row refers to the population scenario year simulated by WATHNET. The second row reports the number of evaluations required to obtain the near-optimal result for each population scenario. The third row reports the elapsed time in hours using the 10-node Athlon PVM cluster. The last row reports the estimated elapsed time to obtain the same result using a single Athlon processor. The estimated time was based on the observation that a speedup of up to 9.93 can be achieved when simulating 1000 50-year replicates for the Sydney headworks system on the 10-node Athlon-based parallel system.

Table 8-2 reveals that a maximum of 5.44 hours elapsed time was required to obtain a near-optimal result for 1000 replicates using the 10-node parallel GA. Without PVM a single processor would run about ten times longer. Clearly this is a significant breakthrough since excessive computing requirements have been a major deterrent to applying these optimisation methods to large-scale complex water supply systems.

<table>
<thead>
<tr>
<th>Population scenario year</th>
<th>2000</th>
<th>2010</th>
<th>2020</th>
<th>2030</th>
<th>2040</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA evaluations</td>
<td>1040</td>
<td>800</td>
<td>800</td>
<td>640</td>
<td>960</td>
</tr>
<tr>
<td>Elapsed time on 10-node Athlon cluster (hr)</td>
<td>5.44</td>
<td>4.25</td>
<td>4.25</td>
<td>3.53</td>
<td>5.34</td>
</tr>
<tr>
<td>Estimated elapsed time using a single Athlon processor (hr)</td>
<td>54.0</td>
<td>42.2</td>
<td>42.2</td>
<td>35.0</td>
<td>53.0</td>
</tr>
</tbody>
</table>
headworks systems. The results demonstrate that parallel computing is a powerful tool for improving computational efficiency.

Note that the number of GA evaluations carried out for each planning year ranged from 640 to 1040. They are quite small compared to other GA optimisation work (e.g. Dandy et al. 1996) where something between 80000 and 10000 was required to find near-optimal solutions. This will be further discussed in the following sections.

8.4 RESULTS FOR REPLICATE COMPRESSION

Another important objective of this study is to investigate the replicate compression method to further improve the efficiency of the Monte Carlo simulation. Replicate compression (RC) was presented in detail in Chapter 5. The method was implemented and embedded into WATHNET. This section presents the results to demonstrate the applicability of the RC method.

Recall that replicate compression can only be used if the objective function depends on penalties arising from restrictions. Therefore, to assess the replicate compression method, the decision variables to be optimised are restricted to the four restriction trigger levels. Because the pump marks are not optimised in this case, they have to be implicitly coded as a part of the default setting of the EDNET program. The pump mark for Warragamba reservoir was set to 90%, and the pump mark for Avon reservoir was 65%. For each population scenario, the GA was run twice, once with replicate compression on and once without replicate compression.

8.4.1 Results

Table 8-3 presents the near-optimal restriction trigger levels obtained from the parallel GA for the five population scenario years with replicate compression applied. Table 8-4 presents the same results without replicate compression. Comparison shows that the near-optimal restriction levels are generally consistent. When the water demand is relatively low, the headworks system can supply sufficient water to the demand area, and there is virtually no need to restrict water
Table 8-3 Near-optimal four restriction levels obtained with replicate compression

<table>
<thead>
<tr>
<th>Population scenario year</th>
<th>Restriction trigger levels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td>10</td>
</tr>
<tr>
<td>2010</td>
<td>10</td>
</tr>
<tr>
<td>2020</td>
<td>49</td>
</tr>
<tr>
<td>2030</td>
<td>52</td>
</tr>
<tr>
<td>2040</td>
<td>57</td>
</tr>
</tbody>
</table>

Table 8-4 Near-optimal four restriction levels obtained without replicate compression

<table>
<thead>
<tr>
<th>Population scenario year</th>
<th>Restriction trigger levels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td>11</td>
</tr>
<tr>
<td>2010</td>
<td>12</td>
</tr>
<tr>
<td>2020</td>
<td>50</td>
</tr>
<tr>
<td>2030</td>
<td>54</td>
</tr>
<tr>
<td>2040</td>
<td>56</td>
</tr>
</tbody>
</table>
consumption. The restriction trigger levels are therefore found consistently at the minimum permitted level of 10% for the years 2000 and 2010 - it is noted that the Warragamba pump mark is quite high. With the increase of water demand (e.g. years 2020, 2030, and 2040) the near-optimal restriction levels gradually climb.

Table 8-5 compares the penalty results due to imposing water restrictions for the five population scenario years. The penalty for imposing water restrictions gradually increases with the increase of water demand. It is clear that the penalty results are quite consistent when using or not using replicate compression, suggesting the application of replicate compression is successful.

Table 8-6 compares the computational turnaround time for a single objective function evaluation with or without replicate compression. The average turnaround time was about 19 seconds/evaluation when RC approach was not applied and ranged from 3 to 10 seconds/evaluation when RC was used.

Table 8-5 Comparison of water restriction expected costs

<table>
<thead>
<tr>
<th>Population scenario year</th>
<th>Restrictions penalty ($/year)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without RC</td>
</tr>
<tr>
<td>2000</td>
<td>801</td>
</tr>
<tr>
<td>2010</td>
<td>8936</td>
</tr>
<tr>
<td>2020</td>
<td>251413</td>
</tr>
<tr>
<td>2030</td>
<td>1207528</td>
</tr>
<tr>
<td>2040</td>
<td>2736225</td>
</tr>
</tbody>
</table>
Table 8-6 Comparison of the computational turnaround time on the
10-node Athlon PVM cluster (Second/evaluation)

<table>
<thead>
<tr>
<th>Replicate compression</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>on</td>
<td>3.0</td>
</tr>
<tr>
<td>off</td>
<td>18.75</td>
</tr>
</tbody>
</table>

8.4.2 Discussion

As outlined in Chapter 6, SIMNET is used to simulate the operation of the headworks system one season at a time using information about streamflow, demand and predefined demand restriction rules, etc. It creates an unformatted network output file, which stores results in a compressed form. WATOUT can then be used to examine the contents of this file by displaying extensive graphical and tabular results. The near-optimal restriction trigger levels for each population scenario have been obtained by running the parallel GA. To help interpret the results, the near-optimal restriction levels for the year 2040 were simulated on the PC version of WATHNET in order to visualise the results using WATOUT.

Figure 8.1 shows how to edit restriction levels in SIMNET. Figure 8.2 illustrates total system storage traces for selected percentiles produced by WATOUT. It shows that one of the lowest system storages occurred at simulation step 460, replicate 860 and year 2033. Since water restrictions are more likely to be imposed when the system storage is low, the results for replicate 860 are used to provide insight about system performance.
Demand group: 1
First demand node in group: Sydney Enhouse

Rule applies from year 1998 to year 2049 with last year being 2049

Zone 1 - Zone 2 - Zone 3 - Zone 4
Restriction: |---|---|---|---|
Storage: 56 56 56 51 0

- Revise restrictions at start of every season
- Announce restriction level at start of season, after which relax level until end of season if storage improves, and lift restrictions at end of season.
Note level cannot be made more severe after first announcement

Figure 8.1 Example of restriction rule set-up in SIMNET

System lowest volume & percentiles
4,99

Figure 8.2 Time series trace for the lowest, 4 and 99 percentile total system storage
Figures 8.3, 8.4, and 8.5 present the simulation results for the population scenario years 2000, 2020, and 2040 respectively for replicate 860. The curve in the top part of each figure represents the total system storage variation during the simulation time period. The dashed line is the trigger storage level obtained from the GA. The bottom part of each figure illustrates the simulation period(s) experiencing demand shortfall. Recall that in the GA, each set of decision variables has to be evaluated using WATHNET simulation model to perform Monte Carlo analysis for 1000 replicates. If replicate compression is not applied, each replicate has to be evaluated by WATHNET for the entire simulation time period of 600 steps, that is, 50 simulation years in monthly intervals. This is why the turnaround time per evaluation is nearly constant for each different GA run when replicate compression was not applied.

However, when replicate compression is applied the critical periods for each replicate are identified first and the objective function is only evaluated for such periods. As outlined in Chapter 5, the critical period starts when a full system commences a drawdown, which passes through the restriction trigger level and ends when the system passes through the trigger level during its recovery. If the restriction trigger level is low (eg. Figure 8.3 for year 2000) and there is no critical period identified for that replicate, the Monte Carlo simulation for the entire replicate can be skipped. With the increase of demand, the restriction trigger levels increase. Therefore, the number and duration of critical periods requiring simulation also increase. This explains why the turnaround time increases with demand when replicate compression is applied.

From the results presented above, it is clear that the results with or without the replicate compression are overall consistent. Table 8-6 reveals that, for each evaluation using 1000 replicates, the turnaround time using replicate compression is about 6 times faster for scenario year 2000, and about 2 times faster for scenario year 2040. It is therefore concluded that the power of replicate compression can further improve the Monte Carlo simulation effort particularly if the system is operated with a low probability of imposing restrictions.
Figure 8.3 System performance for population scenario year 2000: replicate 860
Figure 8.4 System performance for population scenario year 2020: replicate 860
Figure 8.5 System performance for population scenario year 2040: replicate 860
Unfortunately, this conclusion must be tempered by the following limitation. As discussed in Chapter 5, if the objective function only depends on restriction penalties, it can be evaluated only for the replicates, which experience one or more critical periods where system storage falls below the trigger level. Therefore, the replicate compression approach is limited to the applications that focus exclusively on optimising restriction levels.

8.5 RESULTS FOR OPTIMISING EIGHT DECISION VARIABLES

8.5.1 Optimisation Results Based on 1000 Replicate Simulations

The principal goal of this study is to explore the use of modern probabilistic search strategies to find optimal solutions in complex water resource systems. This section culminates this quest by examining the results from optimising eight decision variables that affect the performance of the Sydney headworks system. The eight decision variables include four restriction trigger levels, two pump marks for the two demand areas and two desalination plant variables.

Table 8-7 presents the eight decision variables optimised by the parallel GA for five different population scenarios. The first four variables are used to derive the four trigger levels using equation (7.12). The fifth and sixth values are used to calculate the pump marks for the Warragamba and Avon reservoirs. The last two are used to calculate the desalination options. The eight decision variables shown in Table 8-7 will be further interpreted using Figures 8.6, 8.7 and Table 8-8.

Figure 8.6 shows the near-optimal restriction trigger levels. The general trend is that the restriction trigger levels increase with the increase of water demand due to population growth. However, the four levels are not well spread out in the search range; instead, they are close to each other. This issue will be discussed in the next section.
Table 8-7 Eight decision variables obtained from the GA for different population scenario years

<table>
<thead>
<tr>
<th>Population scenario year</th>
<th>Decisions obtained from the GA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2000</td>
<td>0.018</td>
</tr>
<tr>
<td>2010</td>
<td>0.122</td>
</tr>
<tr>
<td>2020</td>
<td>0.007</td>
</tr>
<tr>
<td>2030</td>
<td>0.177</td>
</tr>
<tr>
<td>2040</td>
<td>0.048</td>
</tr>
</tbody>
</table>

Figure 8.7 shows the two near-optimal pump marks obtained by the GA. It is observed that the pump mark for Warragamba storage generally increases with population growth. From population scenario years 2000 to 2040, the pump mark increases from 45% to 68% for the Warragamba storage. The pump mark for Avon reservoir shows a similar consistency with population growth, except for year 2030 where an unexpectedly low pump mark was obtained.

Table 8-8 presents the optimal results for the desalination option. The trigger level for desalination plant is, by constraint, lower than the lowest trigger level for outdoor water restrictions in order that the desalination plant would be the last option to be chosen for supplying water. By the year 2040, the parallel GA was opting for the largest desalination plant permitted to be built if Warragamba storage falls below 27%.

Table 8-9 presents a breakdown of the objective function components for each population scenario year. Rows 1 and 2 are the penalties of imposing restrictions for
Figure 8.6 Near-optimal restriction trigger levels for each scenario year

Figure 8.7 Near-optimal pump marks for each scenario year
Table 8-8 Desalination results obtained from the GA

<table>
<thead>
<tr>
<th>Decisions</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td><strong>Trigger level</strong></td>
<td></td>
</tr>
<tr>
<td>(% Warragamba storage)</td>
<td>1</td>
</tr>
<tr>
<td><strong>DesalCap (ML/day)</strong></td>
<td>170</td>
</tr>
</tbody>
</table>

Table 8-9 Objective function component penalties for optimising eight decision variables ($1000/year)

<table>
<thead>
<tr>
<th>Operating option</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>Restriction – outdoor (1)</td>
<td>32.2</td>
</tr>
<tr>
<td>Restriction – commercial+indoor (2)</td>
<td>0.0</td>
</tr>
<tr>
<td>Pump- Warragamba (3)</td>
<td>1128.8</td>
</tr>
<tr>
<td>Pump- Avon (4)</td>
<td>77.9</td>
</tr>
<tr>
<td>Desalination-construction (5)</td>
<td>0.0</td>
</tr>
<tr>
<td>Desalination-production (6)</td>
<td>0.0</td>
</tr>
<tr>
<td>Reliability constraint penalty (7)</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>Total penalty (8)</strong></td>
<td>1238.9</td>
</tr>
</tbody>
</table>
outdoor and commercial/indoor water use. Rows 3 and 4 represent the penalties of pumping for the Warragamba and Avon reservoirs respectively. Row 5 is the cost of constructing the desalination plant and row 6 is the cost of producing water from the desalination plant. Row 7 is the penalty of not meeting the minimum system reliability of 90%. The last row is the total penalty for each population scenario year. It is observed that the general trend for each penalty is that it increases with water demand. Of significance here is the increase of the penalty due to imposing water restrictions. For year 2000 there is no penalty incurred for constructing the desalination plant and utilising its water – this means the desalination plant was not commissioned. However, for the year 2010 onward, the desalination plant, when Warragamba storage falls below the trigger level, is commissioned with increasing capacity. Although the desalination option is activated from year 2010, the water usage from desalination is significantly very small. It is also shown that no penalty is incurred for failure to satisfy the minimum reliability of 90% for all population scenario years.

Figure 8.8 shows the percentage distribution of the objective function component penalties for water restrictions on outdoor and indoor/commercial water, pumping from the Shoalhaven system, and constructing and producing water from the desalination plant. For the year 2000 when the demand is relatively low, the total penalty of the system is primarily due to pumping, which accounts for about 97% of the total penalty. For years 2010 to 2040, however, the penalty of imposing water restrictions increases from 13% to 65% of the total penalty value, whilst the penalty of pumping drops from 87% to 34% of the total penalty. Note that the desalination cost is not clearly shown on the Figure 8.8 due to its small value. However, the penalty of utilising desalination water increases from 0.23% to 0.54% for years 2010 to 2040.

8.5.2 Discussion

Table 8-10 presents the average annual demand for each population scenario over 1000 replicates. This will assist with the discussion of the results, the objective of this section.
Table 8-11 presents indicative unit costs ($/ML) associated with each operating option. The cost of different levels of restrictions are derived using equation (7.5) and the existing restriction policy described in Table 6.11. The indicative cost of rationing indoor/commercial water is based on the monthly GDP of the Sydney region of $12.5 billion. The cost of constructing and operating the desalination plant is based on Linstrum et al’s study (2000). Table 8-11 clearly shows that pumping from the Shoalhaven system is the cheapest ($20/ML) of all the options, while using water from a desalination plant ($2,000,000 per ML/day installed capacity plus $2000/ML) or imposing restrictions on indoor or commercial use should be options of last resort.
As shown in Table 8-9 and Figure 8.8, when the demand is low, the GA opts for the cheapest option to supply the system. In such a case, the GA would first choose to pump water from the Shoalhaven to supply the system. Therefore, pumping was the major option that the GA preferred for water supply for years 2000 and 2010. The pump marks for Warragamba and Avon reservoirs increase gradually to meet the demand from the population scenario years 2000 to 2040. For the year 2040 (Figure 8.7), relatively high pump marks for the Warragamba (68%) and Avon reservoirs

<table>
<thead>
<tr>
<th>Population scenario year</th>
<th>2000</th>
<th>2010</th>
<th>2020</th>
<th>2030</th>
<th>2040</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average annual demand (ML)</td>
<td>571390</td>
<td>620600</td>
<td>663300</td>
<td>700000</td>
<td>724910</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outdoor restriction- Level</th>
<th>125</th>
<th>Rationing commercial/indoor</th>
<th>250000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outdoor restriction – Level 2</td>
<td>185</td>
<td>Pumping</td>
<td>20</td>
</tr>
<tr>
<td>Outdoor restriction – Level 3</td>
<td>305</td>
<td>Desalination construction cost</td>
<td>2000000 ($/ML/day installed capacity)</td>
</tr>
<tr>
<td>Outdoor restriction – Level 4</td>
<td>425</td>
<td>Desalination Production cost</td>
<td>2000</td>
</tr>
</tbody>
</table>
(68%) were found by the GA. In addition, the GA choose to increase outdoor restriction trigger levels gradually to meet the increasing demand since it is cheaper to apply water restrictions than to use desalination water. For this reason, the desalination option is minimally employed in the population scenario years 2010, 2020, 2030 and 2040. Overall, the results are judged to be consistent with engineering judgement.

Although it is cheaper to use water pumped from the Shoalhaven system, Figure 8.7 reveals that the maximum pump mark of 100% was not selected by the GA as demand increased – the maximum value was around 76% and 68% for the population scenario years 2030 and 2040. This requires explanation.

As the pump mark increases, the air space in the reservoir available to store a large unregulated inflow is reduced. This increases the chance of a spill from the reservoir. In the case of Avon and Warragamba reservoirs spills are lost to the system. Therefore, as the pump mark is increased more of the pumped water is ‘in effect’ being wastefully pumped over the spillway. For this reason, the GA limits the pump mark. The differing pump marks for the Avon and Warragamba reflect the different reservoir volumes and catchment yields.

It is noted in Figure 8.7 that the pump mark for Avon reservoir does not show consistency with population growth – in particular for year 2030 the pump mark is unexpectedly low. The reasons for this are not clear. It is possible that the GA has not converged. It is noted that Avon only supplies 10% of the population. As a results the Warragamba pump cost dwarfs the Avon cost. It is also possible that the GA has overstated the Warragamba pump mark and to maintain a low overall pumping cost has understated the Avon pump mark – see Figure 8.7.

Table 8-11 can also help explain why the restriction trigger levels are not well spread out in the search range as shown in Tables 8.3, 8.4 and Figure 8.6 and particularly when demand is higher. As demand increases, higher pump marks are used due to their lower cost, while the restriction trigger levels were also found to increase. If the fourth restriction level is too low, the risk of requiring the
desalination option or restricting commercial/indoor water use increases, which will cause a large jump in the objective function. To avoid this, the GA pushes the fourth restriction level close to the first level, so that the restriction option for outdoor water use can be used as fully as possible to reduce the chance of using the expensive desalination option or restricting commercial/indoor water use.

Figures 8.9 and 8.10 present two examples showing the annual system failure probability where failure means imposition of restrictions. They represent the results for the population scenario years 2000 and 2040 respectively. It is clear that the risk of failure increases with increasing water demand and higher restriction trigger levels. The system reliability ranges from 99.5% to 95% from years 2000 to 2040.

Figures 8.11 and 8.12 present two examples showing the annual shortfall probability for Sydney Metropolitan outdoor water use for population scenario years 2000 and 2040 respectively. Figures 8.13 and 8.14 present the annual shortfall probabilities for commercial water use for population scenario years 2000 and 2040 for the Sydney Metropolitan area. The annual shortfall probabilities for indoor water use present the same pattern as for commercial water use and therefore are not shown. Whereas probability of outdoor restrictions climbs to 4 to 5% by year 2040, the probability of commercial restrictions remains virtually zero. This reflects the GA’s avoidance of decisions that trigger high penalty restriction events.

The results for the South Illawarra area present the same pattern as for the Sydney Metropolitan area and therefore are not presented.
Figure 8.9 System annual shortfall probability for year 2000

Figure 8.10 System annual shortfall probability for year 2040
Figure 8.11 Annual shortfall probability for outdoor water use (year 2000)

Figure 8.12 Annual shortfall probability for outdoor water use (year 2040)
Figure 8.13 Annual shortfall probability for commercial water use (year 2000)

Figure 8.14 Annual shortfall probability for commercial water use (year 2040)
8.5.3 Sensitivity of Results to the Number of Replicates

In the previous sections, the results for optimising the operating policy of the Sydney headworks system were presented and discussed. For those results, 1000 replicates were used to simulate the Sydney headworks system using the WATHNET model. However, as mentioned in Chapter 4, a large number of hydro-climatic replicates are required for statistically meaningful risk estimates of system performance for complex urban water supply systems with high levels of reliability. For example, to estimate with 95% confidence the probability of water shortages to within ±0.01 of the true value of 0.05 requires over 1,800 replicates – if the desired accuracy of system reliability is increased, the number of additional replicates can be substantial.

To appreciate the influence of the number of replicates on the optimised results, this section considers the optimisation of the headworks system using 2000 and 10000 replicates respectively.

Tables 8.12 and 8.13 present the penalty results for each operating option using 2000 and 10000 replicates. The general trend for each option is quite consistent with the results for 1000 replicates shown in Table 8-9. For years 2000 and 2010 with low demand, pumping dominates the total penalty. The penalty due to water restrictions on outdoor water use increases with increasing demand. For year 2000 no penalty is incurred for utilising desalination water, although the decision was made by the GA to build the desalination plant for the 10000 replicate optimisation. The desalination plant is commissioned for year 2010 onwards for all replicate runs. However, the water usage from the desalination plant is very small probably because the severe droughts broke not long after the plant was commissioned and lower cost water became available. It is also shown that no penalty is incurred for failure to satisfy the minimum reliability of 90% for all population scenarios.

Table 8-14 shows the decision variables obtained from the GA for all population scenario years and for replicates 1000, 2000, and 10000. Overall, the first four
Table 8-12 Penalty results for 2000 replicates  ($1000/year)

<table>
<thead>
<tr>
<th>Operating option</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>Restriction – outdoor (1)</td>
<td>19.9</td>
</tr>
<tr>
<td>Restriction – commercial+indoor (2)</td>
<td>0.0</td>
</tr>
<tr>
<td>Pump- Warragamba (3)</td>
<td>1128.4</td>
</tr>
<tr>
<td>Pump- Avon (4)</td>
<td>78.7</td>
</tr>
<tr>
<td>Desalination-construction (5)</td>
<td>0.0</td>
</tr>
<tr>
<td>Desalination-production (6)</td>
<td>0.0</td>
</tr>
<tr>
<td>Reliability constraint penalty (7)</td>
<td>0.0</td>
</tr>
<tr>
<td>Total penalty (8)</td>
<td>1227.0</td>
</tr>
</tbody>
</table>

Table 8-13 Penalty results for 10000 replicates  ($1000/year)

<table>
<thead>
<tr>
<th>Operating option</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>Restriction – outdoor (1)</td>
<td>191.1</td>
</tr>
<tr>
<td>Restriction - commercial+indoor (2)</td>
<td>0.0</td>
</tr>
<tr>
<td>Pump- Warragamba (3)</td>
<td>611.1</td>
</tr>
<tr>
<td>Pump- Avon (4)</td>
<td>69.9</td>
</tr>
<tr>
<td>Desalination-construction (5)</td>
<td>9.7</td>
</tr>
<tr>
<td>Desalination-production (6)</td>
<td>0.0</td>
</tr>
<tr>
<td>Reliability constraint penalty (7)</td>
<td>0.0</td>
</tr>
<tr>
<td>Total penalty (8)</td>
<td>881.8</td>
</tr>
</tbody>
</table>
Table 8-14 Decision variables for each replicate run and population scenario year

<table>
<thead>
<tr>
<th>Year</th>
<th>Number of replicates</th>
<th>Restriction trigger levels</th>
<th>Pump marks</th>
<th>Desalination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2000</td>
<td>1000</td>
<td>0.018</td>
<td>0.012</td>
<td>0.054</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.006</td>
<td>0.009</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.123</td>
<td>0.029</td>
<td>0.054</td>
</tr>
<tr>
<td>2010</td>
<td>1000</td>
<td>0.122</td>
<td>0.013</td>
<td>0.052</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.011</td>
<td>0.012</td>
<td>0.052</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.037</td>
<td>0.008</td>
<td>0.059</td>
</tr>
<tr>
<td>2020</td>
<td>1000</td>
<td>0.007</td>
<td>0.005</td>
<td>0.034</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.057</td>
<td>0.006</td>
<td>0.185</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.022</td>
<td>0.003</td>
<td>0.061</td>
</tr>
<tr>
<td>2030</td>
<td>1000</td>
<td>0.177</td>
<td>0.043</td>
<td>0.173</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.105</td>
<td>0.011</td>
<td>0.035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.029</td>
<td>0.073</td>
<td>0.009</td>
</tr>
<tr>
<td>2040</td>
<td>1000</td>
<td>0.048</td>
<td>0.106</td>
<td>0.133</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.001</td>
<td>0.085</td>
<td>0.006</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.064</td>
<td>0.039</td>
<td>0.094</td>
</tr>
</tbody>
</table>

Decision variables for restriction trigger levels are consistent regardless of the replicate run for each population scenario year. However, there are exceptions. For
example, for year 2000 and 10000 replicates, the fourth restriction level and Warragamba pump mark appear to be anomalous. In fact the GA has reversed the high pump mark and low restriction trigger level strategy to find a solution with a lower expected annual penalty than for the 1000 and 2000 replicates runs.

The pump marks show overall consistent behaviour, though once again some large variations are noted. The Avon pump mark exhibits more variability than the Warragamba pump mark for a given population scenario year. This is attributed to the fact that the Warragamba pumping cost dominates overall pumping cost - this is to be expected as Avon reservoir only serves 10% of the population.

The desalination decisions show considerable variability. This is to be expected given that desalination minimally contributes to the overall penalty.

Table 8-15 summarises the total penalty value for each replicate run and population scenario year. It suggests that for each population scenario year, the total penalty value obtained from the GA for the three different replicate runs appears to be consistent. To interpret the difference, Table 8-16 presents the standard error (STDEV) of the total penalty for each replicate run. It is seen that for a given replicate run, the STDEV increases with population scenario year. For year 2000, the STDEV values for 1000 and 2000 replicates are consistently small. Tables 8.9 and 8.12 show that pumping from the Shoalhaven system is the dominant water supply option accounting for nearly 99% of total system penalty. The regular occurrence of pumping during the simulation time period is offered as a possible reason why the STDEV is very low for year 2000. However, with increase of water demand, water restrictions will be imposed more frequently accounting for around 10% of the total penalty for year 2010 and 65% of the total penalty for year 2040. As water restrictions become more frequent, they introduce greater variability in the expected penalty. This helps explain the high the STDEV value for the year 2040 for 1000-replicate run.
Table 8-15 Total penalty value for each replicate run and population scenario year ($1000/ML)

<table>
<thead>
<tr>
<th>Number of replicates</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
<td>1238.9</td>
</tr>
<tr>
<td>2000</td>
<td>1227.0</td>
</tr>
<tr>
<td>10000</td>
<td>881.8</td>
</tr>
</tbody>
</table>

Table 8-16 also demonstrates that STDEV decreases as the number of replicates increases. This is to be expected because STDEV is inversely proportional to the square root of the number of replicates. The results in Table 8-16 are consistent with this relationship except for the year-2000 10000-replicate run which will be discussed below.

Table 8-17 presents the 95% confidence interval on the total penalty using \( TOTPEN \pm 1.96 \times STDEV \), where the \( TOTPEN \) is the total penalty from the GA run. With the exception of the year-2000 10000-replicate run, the confidence intervals overlap indicating that the GA results are consistent over different replicate runs. Table 8-17 makes clear that use of 10000 replicates produces the most accurate estimates and presumably the most accurate decisions.
Table 8-16 Standard error of penalty (STDEV) for each replicate run and population scenario year ($1000/ML)

<table>
<thead>
<tr>
<th>Number of replicates</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
<td>10.6</td>
</tr>
<tr>
<td>2000</td>
<td>7.4</td>
</tr>
<tr>
<td>10000</td>
<td>9.1</td>
</tr>
</tbody>
</table>

Table 8-17 95% Confidence interval on the total penalty

<table>
<thead>
<tr>
<th>Number of replicates</th>
<th>Population scenario year</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
<td>1218 - 1260</td>
</tr>
<tr>
<td>2000</td>
<td>1219 - 1242</td>
</tr>
<tr>
<td>10000</td>
<td>863 - 900</td>
</tr>
</tbody>
</table>
The GA result for the year-2000 10000-replicate run needs further comment. In Table 8-13, the penalty value found by the GA for year 2000 is significantly lower than the penalty values for the 1000 and 2000 replicate runs. Table 8-14 reveals that the GA found a distinctly different set of decisions for the 10000 replicate run. As already noted, pumping was the major option accounting for almost 99% of the total operating cost for year 2000 when 1000 and 2000 replicates were used. However, the GA solution for the 10000 replicate run opted for a different strategy to minimise the total expected penalty, namely greater reliance on outdoor water restrictions and lesser reliance on pumping. It is apparent that the GA converged to a better solution in the 10000 replicate run. Certainly, the expected annual penalty is significantly lower. Moreover, for the other population scenario years consistent results were obtained by the GA for all replicate runs suggesting the GA converged prematurely for the 1000 and 2000 replicate runs for year 2000.

8.6 CONCLUSION

This Chapter presented the results for optimisation of the Sydney headworks systems for five population scenarios representing five different levels of demand. Because of simplifications to the headworks system, the results are at best indicative. However, the primary objective was to demonstrate the feasibility of parallel GA.

It was demonstrated that the parallel GA operating on a cluster of processors under the PVM protocol is a feasible tool for which the speedup was nearly proportional to the number of processors employed. It is suggested that parallel GA has the potential to optimise complex water resources systems in a practical and realistic manner.

A discussion of the results on replicate compression suggested that replicate compression can significantly reduce the computational turnaround time for Monte Carlo simulation. Unfortunately this conclusion must be tempered by the limitation that the objective function must depend on penalties arising from restrictions only.

Although it cannot be established that the GA converged to a true optimum, the soundness of the near-optimal solutions can be assessed by considering the unit costs
associated with each operating option. The pump option was found dominant when the demand was low. Trigger levels for restrictions on outdoor water use were found to increase with increasing demand, while the expensive desalination option was largely avoided by the GA. Interestingly, the GA tended to opt for restriction rules which rapidly progressed to a complete ban on outdoor water use. This minimised the chance of requiring desalination or imposing restrictions of commercial/indoor use. The GA avoided high pump marks because much of the pumped water would have been wasted. Overall the GA results were judged sound in accordance with engineering judgement.

The sensitivity of the optimised results to the number of replicates was investigated. Comparison of results for 1000, 2000 and 10000 replicates revealed overall consistency in penalties with an exception where it appears the GA prematurely converged. Likewise the GA decisions were largely consistent with decisions associated with the smallest contributions to the total penalty showing greatest variations.

In conclusion, the Sydney headworks case study has demonstrated the feasibility of parallel GA to identify near-optimal solutions for a complex system subject to stochastic forcing. It has also highlighted issues with regard to the number of replicates used in the Monte Carlo simulation and the convergence of GA.
CHAPTER 9

CONCLUSIONS AND
RECOMMENDATIONS
9.1 INTRODUCTION

In the introductory Chapter it was stated that the primary objective of this thesis was to demonstrate the application of probabilistic search methods in conjunction with Monte Carlo simulation to solve complex water resource problems characterised by nonlinearity, input stochasticity, and complexity. The second objective was to explore the benefit of high performance computing to make the optimisation computationally feasible. This chapter will assess how well these objectives were realised. It presents the major conclusions and findings of the thesis and outlines future research directions.

9.2 SUMMARY AND CONCLUSIONS

9.2.1 Recent Advances in Optimisation Models

The review of optimisation in water resource systems concluded that traditional optimisation models, such as linear, nonlinear, and dynamic programming, have only been successfully applied to systems with known future inputs and/or low levels of complexity. None of the traditional optimisation approaches found in the literature were able to cope comprehensively with the following characteristics of water resource planning and operation problems: nonlinearity, stochastic inputs, and complexity. Modern stochastic optimisation approaches have an advantage over traditional methods because they better accommodate the characteristics of water resources systems. Genetic algorithms (GAs) and the shuffled complex evolution algorithm (SCE) are two widely used stochastic approaches. However, there are few applications of such optimisation techniques to water supply headworks systems. The technology used to plan and operate water supply headworks systems remains wanting in its ability to identify good or near-optimal solutions. This provided the motivation for developing a robust and efficient optimisation model that can be applied to solve complex water resource planning and management problems.

9.2.2 Modern Probabilistic Search Methods

The GA represents a class of evolutionary search algorithms. It has been used in a wide spectrum of problems in different fields. Chapter 3 described adaptations to the
standard GA. For the GA to be robust (i.e. avoid premature convergence and locate the global optimum), two lesser-known genetic operators, population selection strategy (PSS) and inversion were employed. The PSS strategy ensures the preservation of good genetic material from the parent population thereby allowing the crossover operator to concentrate on maintaining diversity. With this strategy in place, a very high crossover probability can be used to vigorously explore the decision space, which maximises the chance of avoiding pre-mature convergence on possibly extensive flat regions characteristic of objective functions associated with Monte Carlo simulation of water supply headworks systems.

The SCE method is another promising search method. It conducts multiple concurrent searches within the parameter space. Each search is based on a complex of parameters, initially sampled from the search space. At each iteration, a simplex is randomly selected from each complex and allowed to evolve using a probabilistic variant of Nelder and Mead’s simplex search. The robustness of the algorithm comes from the periodic shuffling of the complex which substantially improves the chance of locating the global optimum. In this study, the SCE method described by Duan et al (1992) was adopted and described in detail in Chapter 3.

9.2.3 Robustness and Efficiency of the GA and SCE Methods

Realistic optimisation of the operation and planning of urban water supply headworks systems requires that the issues of complexity and stochastic forcing be addressed. The only reliable way of accomplishing this is to use simulation models in conjunction with the Monte Carlo method which generates multiple hydro-climate replicates. Probabilistic search methods can be coupled to a generalised simulation model and accommodate complexity as well as stochastic inputs.

However, the main deterrent to such studies is the massive computational burden. Before committing substantial super-computing resources to a real large-scale optimisation problem, a simple case study, deemed sufficiently rich to identify the main challenges but sufficiently simple to be computationally manageable, was developed in Chapter 4. The case study provided insight about objective functions
typical of those to be encountered when optimising a system whose performance is to be evaluated using Monte Carlo simulation. It was found that the objective function surface over which the searches were conducted differed markedly from the smooth surfaces one typically encounters. It was characterised by many piecewise-flat regions separated by steep steps. This suggested that traditional gradient-based methods that rely on gradient information are likely to be ineffective – once located on any extensive flat region such search methods are likely to terminate their searches, most likely prematurely. Probabilistic search methods use randomised rules to guide the search and thus, in principle, possess the ability to negotiate flat regions. Therefore, particular attention was directed at the efficiency and robustness of the GA and SCE search methods in this simple case study.

Both the GA and SCE methods were found well suited to cope with the piecewise flat objective function surface characteristic of the headworks optimisation problem because they have the inherent capability of exploring beyond the extent of a flat region. Both methods were capable of reliably locating near-optimal solutions in such complex problems with stochastic forcing. The SCE appeared to be computationally more efficient than the GA requiring up to 25% fewer evaluations. The SCE method is recommended especially when fast location of a good solution is desired. However, optimisation of complex urban headworks systems is usually so computationally demanding that super-computing resources must be used. Whereas parallelism is inherent in the GA, the SCE method offers limited opportunity for parallel computing.

### 9.2.4 Implementation of Parallel GA and Replicate Compression

This thesis explored two approaches to improve computational efficiency. The first approach was to explore the benefits of parallel computing and the ability of probabilistic search methods to exploit parallelism. One of the main advantages of the GA over other probabilistic search methods is its inherent parallelism. In a parallel-computing environment, the objective function evaluation for the entire population can be performed concurrently because each individual is independent of
the others. However, experience with parallel implementation of the GA method in optimisation of water resource problems is far from complete.

In this study, the customised GA, described in Chapters 3 and 4, has been successfully implemented in a parallel-computing environment using the parallel virtual machine (PVM) as the message passing protocol. PVM is one of the software protocols used in the parallel computation community. It can be programmed in four different approaches depending on the problem to be solved. The master-slave approach was selected for the GA parallelisation. The Pool of Tasks paradigm was used to handle the load-balancing problem for PVM applications to ensure that each host is doing its fair share of work and thus maximising PVM performance.

The second approach, termed replicate compression, was based on the idea of using critical periods. It was implemented to reduce simulation effort by identifying critical periods during which restrictions are to be encountered.

### 9.2.5 Optimisation of Operation of Sydney Water Supply Headworks System

The Sydney headworks system was used as a case study to investigate the key aspects of a full-scale headworks optimisation. This system consists of a network of nine reservoirs, multiple demands and inter-basin pump transfers.

The case study used the parallel GA to find near-optimal solutions according to an objective function, which sought to minimise the total expected operating costs. The decision variables consisted of four restriction trigger levels, two pump marks and two desalination options. For each set of decision variables, the generalised simulation model WATHNET was used to simulate system performance using 1000 50-year replicates of streamflow and demand.

#### 9.2.5.1 The Power of Parallel Computing

A PVM cluster of 10 Athlon-based processors was used to perform the optimisation. A speedup of up to 9.93 was obtained when 10 processors were used. This high efficiency arose because inter-processor communication was negligible when compared with the effort required to perform a 1000 50-year replicate
simulation. It was concluded that the parallel GA utilising a cluster of processors under the PVM protocol operates at close to 100% efficiency.

9.2.5.2 Application of Replicate Compression

Replicate compression was the second approach proposed in this study to reduce the computational turnaround time due to Monte Carlo simulation. A discussion of the results using replicate compression concluded that replicate compression could further improve the computational turnaround time for Monte Carlo simulation particularly if the system is operated with a very low probability of imposing restrictions. Unfortunately this conclusion must be tempered by a major limitation of replicate compression, namely that the objective function must depend on penalties arising from restrictions only.

9.2.5.3 Optimisation of the Sydney Headworks System

The eight decision variables, affecting restrictions, inter-basin transfers and desalination, were optimised for five different population scenario years 2000, 2010, 2020, 2030 and 2040. The near-optimal results obtained by the GA were interpreted by considering the ranking of costs for pumping, restrictions and desalination.

The GA yielded solutions that were consistent with this ranking. It was found that the pump marks and restriction trigger levels increased with increasing water demand. When the demand was low, the objective function was dominated by pumping with the chance of outdoor restrictions minimised by very low restriction trigger levels. However, the restriction storage trigger levels gradually increased with increasing water demand. For year 2040, more than half of the total penalty was due to imposition of outdoor water restrictions even through the expected cost of pumping increased when compared to year 2000. At high levels of demand, the GA largely avoided the desalination option choosing to impose high restriction trigger levels and pump marks.

This study explored the sensitivity of the optimisation problem to the number of replicates used in the Monte Carlo simulation. The results for 1000, 2000, and 10000
replicates were compared. It was shown that the optimised penalties were statistically consistent with the exception of the year 2000 scenario where it is believed that the GA converged prematurely for 1000 and 2000 replicate runs. Likewise, the GA decisions were overall consistent with decisions least affecting the objective function showing greatest variability.

9.3 RECOMMENDATIONS AND FUTURE DIRECTIONS

Despite the promising performance of the parallel GA, more work is required to address methodological shortcomings and explore the capabilities of the parallel GA. This section presents a brief summary of these issues to identify future research directions:

- A major difficulty when using a GA for optimisation lies with the number of parameters that must be specified to control how the decision space is searched (Reed et al 2000). One should adequately size the population of potential designs, set the selection pressure, the probabilities of GA operators such as crossover and mutation, the run length (or the number of generations) so that the GA effectively searches the decision space for a given problem. Several authors (De Jong 1975, Schaffer et al 1989) have tried to find parameter values that work well across a variety of function optimisation problems. However, they have assumed a traditional genetic algorithm, the derived parameter sets, if applied to other models, would probably lead to poor results.

In Chapter 4, the customised GA proved to be efficient and robust in identifying the optimal solution. The GA parameters found in that simple case study were used for the Sydney case study. Although the results from optimising the Sydney headworks system provided valuable insight for understanding the system operating policies, there is no guarantee that the GA parameter settings adopted from Chapter 4 are the best set of GA parameters for the Sydney headworks system.

How to select robust GA parameters to date has been a time-consuming trial-and-error process owing to the large number of possible combinations of parameter
values. For example, 60 runs to identify appropriate parameter values were reported by Aly and Peralta (1999). Future research should be directed at efficient methods to tune the GA parameters for the problem under consideration.

- The Sydney headworks case study represented a relatively small optimisation problem involving only 8 decisions. Reservoir drawdown curves were not optimised, although it is noted that the network linear program embedded in the WATHNET simulation model controls the drawdown according to its own objective function. Considerable work remains to be done investigating the ability of the parallel GA to optimise problems with more decision variables.

- In Sydney case study an objective function based on economic costs was constructed. Recognising that the case study was a demonstration of the parallel GA, several simplifications were made. It is recognised that formulation of an objective function that properly accounts for economic losses incurred by the community during severe drought represents a significant task.

- The Sydney case study considered only a single objective, namely minimising the total expected operating cost. However, many optimisation problems are multi-objective in nature, requiring trade-offs between economic, environmental and social objectives. It is possible to use a single-criterion formulation by imposing constraints on the other criteria, or by incorporating multiple criteria into a single objective function using weighting factors. Neither of these approaches is entirely satisfactory (Halhal et al. 1997).

Since the GA is a multi-start, parallel search algorithm and deals with a pool of solutions at a time, it can be adapted relatively easily to solve multi-objective problems. When applied to multi-objective problems, the general procedure of GA operations and offspring generation remains largely unchanged. The main difference lies with the evaluation of fitness of each solution. For a multi-objective problem, there exists a family of optimal solutions that are known as the Pareto optimal solution set (Goldberg 1989). Certain modifications are therefore required for GAs to find solutions on the trade-off surface of the multiple objectives. A brief literature survey reveals that there are three kinds of multiple-
objective GAs, namely VEGA (Schaffer 1984), SGA (Goldberg 1989) and MOGA (Fonseca and Flemming 1995). Little work has been reported using multi-objective GAs in water supply problems.

9.4 CONCLUDING REMARK

Despite these reservations, this thesis has demonstrated the promise of the parallel GA coupled with Monte Carlo simulation using generalised headworks simulation models. The significance of this work is the development of an optimisation approach that makes substantially fewer compromises about network complexity, nonlinearity and stochastic forcing than previous efforts at optimisation of headworks system.
REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES

Hsu, S. K., Shortage indices for water resources planning in Taiwan, *Journal of Water Resources and Management*, 121(2), March/April 1995.


Martin, Q. W., Optimal reservoir control for hydropower on Colorado River, Texas, Journal of Water Resources Planning and Management, 121(6), November/December 1995.


REFERENCES


REFERENCES


REFERENCES


Perera, B. J. C., Codner, G. P., Reservoir targets for urban water supply systems, *Journal of Water Resources Planning and Management*, 122(4), July/ August 1996


Richardson, J. T., Palmer, MR, Liepins, G. and Hilliard, M., Some guidelines for genetic algorithms with penalty functions, in: J. D. Shaffer, Ed., *Proceedings of The*
REFERENCES


REFERENCES


REFERENCES


## Appendix 1 PVM TERMINOLOGY

This appendix provides a list of PVM terminology. It is intended to assist readers to understand parallel GA implementation in Chapter 5.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer</td>
<td>A temporary storage area in memory. Many methods for routine messages between processors use buffers at the source and destination or at intermediate processors.</td>
</tr>
<tr>
<td>Daemon</td>
<td>A special-purpose process that runs on behalf of the system, for example, the pvmd process or group server task.</td>
</tr>
<tr>
<td>Host</td>
<td>A physical machine, especially a self-complete one on a network with others, for example, Unix workstation or parallel computer</td>
</tr>
<tr>
<td>Message</td>
<td>An ordered list of data sent between tasks.</td>
</tr>
<tr>
<td>Node</td>
<td>Basic computer building block of a multicomputer. Typically a node refers to a processor with a memory system and a mechanism for communicating with other processors in the system.</td>
</tr>
<tr>
<td>Parallel computer</td>
<td>A computer system made up of many identifiable processing units working together in parallel. The term is often used synonymously with concurrent computer to include both multiprocessor and multicomputer. The term concurrent is more commonly used in the United States, whereas the term parallel is more common in Europe.</td>
</tr>
<tr>
<td>Process</td>
<td>A program, data stack, etc... For example, a Unix process or a node program.</td>
</tr>
<tr>
<td>Pvmid</td>
<td>PVM daemon, a process that serves as a message router and virtual machine coordinator. One PVM daemon runs on each host of a virtual machine.</td>
</tr>
<tr>
<td>Spawn</td>
<td>To create a new process or PVM task, possibly different from the parent.</td>
</tr>
<tr>
<td>Task</td>
<td>The smallest unit of a program addressable in PVM. A task is generally a native “process” to the machine on which it runs.</td>
</tr>
<tr>
<td>TID</td>
<td>A unique (parallel virtual machine) identifier associated with each task.</td>
</tr>
<tr>
<td>Virtual machine</td>
<td>Combination of hosts running as a single concurrent computational</td>
</tr>
</tbody>
</table>
This appendix provides some of PVM 3.4 Fortran routines used in implementing parallel GA in Chapter 5.

<table>
<thead>
<tr>
<th>FORTRAN ROUTINES</th>
<th>SYNOPSIS AND PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pvmfspawn( )</strong></td>
<td>Call <code>pvmfspawn(task, flag, where, ntask, tids, numt)</code></td>
</tr>
<tr>
<td>Starts new PVM processes</td>
<td>task - character string containing the executable file name of the PVM process to be started</td>
</tr>
<tr>
<td></td>
<td>flag - integer specifying spawn options eg PvmTaskDefault which means PVM can choose any machine to start task</td>
</tr>
<tr>
<td></td>
<td>where - character string specifying where to start the PVM process /</td>
</tr>
<tr>
<td></td>
<td>ntask - integer specifying the number of copies of the executable to start up.</td>
</tr>
<tr>
<td></td>
<td>tids - array of length at least ntask which contains the tids of the PVM process started by this pvmfspawn call.</td>
</tr>
<tr>
<td></td>
<td>numt - integer returning the actual number of tasks started, values less than 0 indicate a system error. A positive value less than ntask also indicates a partial failure.</td>
</tr>
<tr>
<td>Example</td>
<td>call pvmfspawn(<code>nodeprog', flag, </code>Siouan', 1, tids(3), numt)</td>
</tr>
<tr>
<td><strong>pvmfconfig( )</strong></td>
<td>Call <code>pvmfconfig(nhost, narch, dtid, name, arch, speed, info)</code></td>
</tr>
<tr>
<td>returns information about the present virtual machine configuration.</td>
<td>nhost – integer returning the number of the hosts.</td>
</tr>
<tr>
<td></td>
<td>narch – integer returning the number of different data formats being used.</td>
</tr>
<tr>
<td></td>
<td>dtid - Integer returning pvmd task ID for this host.</td>
</tr>
<tr>
<td></td>
<td>name- character string returning name of this host.</td>
</tr>
<tr>
<td></td>
<td>arch – character string returning name of the host architecture.</td>
</tr>
<tr>
<td></td>
<td>speed- integer returning relative speed of this host.</td>
</tr>
<tr>
<td></td>
<td>info – integer status code returning by the routine. Values less than zero indicates an error.</td>
</tr>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td><strong>Call pvmfconfig (nhost, narch, dtid, name, arch, speed, info)</strong></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>
| **Pvmkill ( )** ter | Call pvmkill(tid, info)  
tid – integer task identifier of the PVM process to be killed.  
info – integer status code returning by the routine. Values less than zero indicates an error. |
|minates a specified  | **Pvmfrecv ( )** re | Call pvmfrecv(tid, msgtag, bufid)  
tid - integer identifier of sending process supplied by the user.  
msgtag - integer message tag supplied by the user, should be >= 0, allows the user's program to distinguish between different kinds of messages. A -1 in this argument matches any msgtag i.e. a wildcard.  
bufid - integer returns the value of the new active receive buffer identifier. Values less than zero indicate an error. |
| PVM process | ceives a message. | Example  
Call pvmfrecv (-1, 4, bufid) info - integer status code returned by the routine, values less than zero indicate an error. |
| **Pvmfinitse | Call pvmfinitse | encoding - specifies the next message's encoding scheme where the options are:  
*PvmDataDefault* ie XDR encoding used by default as PVM cannot know if a user is going to add a heterogeneous machine before this message  
*PvmDataRaw* ie no encoding used when it is known that the receiving machine understands the native format  
*PvmDataInPlace* ie specifies that data be left in place during packing  
bufid - integer returned containing the message buffer identifier. Values less than zero indicate an error. |
| nd () clears daf | encoding - specifies the next message's encoding scheme where the options are:  
*PvmDataDefault* ie XDR encoding used by default as PVM cannot know if a user is going to add a heterogeneous machine before this message  
*PvmDataRaw* ie no encoding used when it is known that the receiving machine understands the native format  
*PvmDataInPlace* ie specifies that data be left in place during packing  
bufid - integer returned containing the message buffer identifier. Values less than zero indicate an error. |
| ult default send | Example  
Call pvmfinitsend (default, bufid)  
clears default send buffers and specifies message encoding. |
| buffers and spe | encoding - specifies the next message's encoding scheme where the options are:  
*PvmDataDefault* ie XDR encoding used by default as PVM cannot know if a user is going to add a heterogeneous machine before this message  
*PvmDataRaw* ie no encoding used when it is known that the receiving machine understands the native format  
*PvmDataInPlace* ie specifies that data be left in place during packing  
bufid - integer returned containing the message buffer identifier. Values less than zero indicate an error. |
| **Pvmfsend ( )** | sends the data in the active message buffer. | call pvmfsend(tid, msgtag, info)  
| tid - integer task identifier of destination process  
| msgtag - integer message tag supplied by the user, should be >= 0.  
| info - integer status code returned by the routine, values less than zero indicate an error.  
| Example  
| Call pvmfsend(tid, msgtag, info) |
| **Pvmfpack ( )** | Packs the active message buffer with arrays of prescribed data type. | Call pvmfpack(what, xp, nitem, stride, info)  
| what - specifies the type of data being packed  
| xp - pointer to the start of a block of bytes, can be of any data type but must match the corresponding unpack data type.  
| nitem - total number of items to be packed (not the number of bytes)  
| stride - the stride to be used when packing the items.  
| info - integer status code returned by the routine, values less than zero indicate an error.  
| Example  
| Call pvmfpack(integer, nitem, 1, 1, info) |
| **Pvmfunpack ( )** | Unpacks the active message buffer into arrays of prescribed data type. | Call pvmfunpack(what, xp, nitem, stride, info)  
| what - specifies the type of data being unpacked  
| xp - pointer to the start of a block of bytes, can be of any data type but must match the corresponding unpack data type.  
| nitem - total number of items to be unpacked.  
| stride - the stride to be used when unpacking the items.  
| info - integer status code returned by the routine, values less than zero indicate an error.  
| Example:  
| Call pvmfunpack(string, stepname, 1, 1, info) |
| **Pvmfexit ( )** | Tells the local pvmd that this process is leaving PVM. This routine doesn’t kill the process. | Call pvmfexit(info)  
| info – integer status code returned by the routine. Values less than zero indicate an error.  
| Example:  
| Call pvmfexit(info) |
This appendix contains total monthly rainfall data (mm) for years 1987 to 1994 in the Newcastle region. It is used to perform a monthly regression analysis between per capita outdoor water consumption and rainfall depth in Chapter 6.

<table>
<thead>
<tr>
<th>Month</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>40.4</td>
<td>11.8</td>
<td>72.9</td>
<td>72.1</td>
<td>97.4</td>
<td>43.4</td>
<td>86.6</td>
<td>99.1</td>
<td>22.9</td>
<td>133.5</td>
<td>136.6</td>
<td>55.0</td>
</tr>
<tr>
<td>1988</td>
<td>94.7</td>
<td>97.2</td>
<td>111.7</td>
<td>332.5</td>
<td>41.6</td>
<td>32.3</td>
<td>51.9</td>
<td>14.7</td>
<td>89.8</td>
<td>4.1</td>
<td>134.3</td>
<td>55.3</td>
</tr>
<tr>
<td>1989</td>
<td>147.5</td>
<td>101.4</td>
<td>118.8</td>
<td>88.9</td>
<td>83.5</td>
<td>336.5</td>
<td>28.4</td>
<td>4.2</td>
<td>16.9</td>
<td>47.2</td>
<td>36.3</td>
<td>47.6</td>
</tr>
<tr>
<td>1990</td>
<td>47.5</td>
<td>531.3</td>
<td>53.9</td>
<td>149.5</td>
<td>82.0</td>
<td>42.6</td>
<td>62.7</td>
<td>114.8</td>
<td>48.3</td>
<td>50.8</td>
<td>12.1</td>
<td>14.3</td>
</tr>
<tr>
<td>1991</td>
<td>64.3</td>
<td>17.4</td>
<td>25.7</td>
<td>1.8</td>
<td>104.9</td>
<td>84.2</td>
<td>50.1</td>
<td>2.5</td>
<td>2.9</td>
<td>24.7</td>
<td>30.2</td>
<td>230.7</td>
</tr>
<tr>
<td>1992</td>
<td>62.8</td>
<td>253.7</td>
<td>187.5</td>
<td>142.7</td>
<td>118.9</td>
<td>34.9</td>
<td>40.0</td>
<td>26.8</td>
<td>37.3</td>
<td>30.4</td>
<td>76.0</td>
<td>142.1</td>
</tr>
<tr>
<td>1993</td>
<td>86.6</td>
<td>45.2</td>
<td>104.4</td>
<td>23.4</td>
<td>47.2</td>
<td>92.4</td>
<td>88.0</td>
<td>68.2</td>
<td>61.8</td>
<td>57.4</td>
<td>81.8</td>
<td>19.4</td>
</tr>
<tr>
<td>1994</td>
<td>31.4</td>
<td>70.0</td>
<td>265.4</td>
<td>58.2</td>
<td>116.4</td>
<td>41.8</td>
<td>60.4</td>
<td>46.6</td>
<td>20.9</td>
<td>43.7</td>
<td>39.7</td>
<td>69.4</td>
</tr>
</tbody>
</table>
This appendix contains the number of rain days per month for years 1987 to 1994 in the Newcastle region. It is used to perform a monthly regression analysis between per capita outdoor water consumption and the number of rain days in Chapter 6.

<table>
<thead>
<tr>
<th>Month</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>12</td>
<td>5</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>14</td>
<td>3</td>
<td>16</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>1988</td>
<td>12</td>
<td>14</td>
<td>17</td>
<td>19</td>
<td>9</td>
<td>11</td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>2</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>1989</td>
<td>15</td>
<td>10</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>1990</td>
<td>14</td>
<td>14</td>
<td>17</td>
<td>11</td>
<td>14</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>1991</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>17</td>
<td>12</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>1992</td>
<td>10</td>
<td>14</td>
<td>17</td>
<td>14</td>
<td>13</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>14</td>
<td>16</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>1993</td>
<td>11</td>
<td>7</td>
<td>17</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>17</td>
<td>8</td>
<td>11</td>
<td>12</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>1994</td>
<td>8</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>13</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>