Multi Reservoir Systems Optimisation Using Genetic Algorithms

Mohammed Sharif

A thesis submitted for the degree of Doctor of Philosophy

The University of Edinburgh

August 1999

Abstract

The operation of multiple reservoir systems for sometimes conflicting purposes can be a complex process because of the involvement of a large number of decision variables and constraints. Dynamic programming (DP) has long been recognised as a powerful approach in the analysis of water resource systems. The usefulness of DP for multi reservoir systems is, however, limited by the huge demand that it can induce on computational resources. Many forms of DP havè been developed to alleviate the problem of dimensionality with varying degrees of success, but no general algorithm exists. This thesis describes the development and application of genetic algorithms (GAs) for the optimisation of multi reservoir systems.

The GA approach is validated through application to a number of problems with known solutions. Several alternative formulations of a GA for reservoir systems are evaluated using the four reservoir problem. This has been done with a view to presenting fundamental guidelines for implementation of the approach to practical problems. Alternative representation, selection, crossover, and mutation schemes are considered. The most promising GA approach comprises real-value coding, tournament selection, uniform crossover, and modified uniform mutation. A non-linear four reservoir problem was also solved, along with a problem with extended time horizons. A more complex ten reservoir problem was also successfully solved.

The practicality of the developed GA approach in the determination of optimal reservoir operating rules is considered through application to a reservoir system in Indonesia. Optimal operating rules have been derived for the existing development situation in the basin, and for two future water resource development scenarios, using critical period hydrology. A comparison of the GA results with those produced by discrete differential DP (DDDP) is also presented. The application of GA approach to real time operations with stochastically generated inflows is also demonstrated for the Equatorial Lakes system in Africa. A methodology for forecasting reliable power that can be produced over different durations of time has also been developed using a GA. For the problems considered in this study, the GA solutions are very close to the optimum. The results demonstrate that the approach is robust and is easily applied to complex systems. It has potential as an alternative to stochastic DP approaches.

Declaration of originality

This thesis was prepared entirely by myself. The work reported here in has been conducted by myself in the School of Civil and Environmental Engineering at the University of Edinburgh, Edinburgh, United Kingdom between October 1996 and August 1999.

Mohammed Sharif

30 August 1999

dedicated to my family

Acknowledgements

My three years of study experience in UK and particularly in Edinburgh was a rewarding one. I have enjoyed every bit of it. I thank all those who tangibly or intangibly contributed to my family's comfortable and enjoyable stay in Edinburgh. I offer my sincere thanks in particular to:

- Dr. Robin Wardlaw, my supervisor, for his advice, guidance, encouragement, and his willingness to share his deep insight during the various stages of this work, and also for going through the thesis;
- the University of Edinburgh for providing me with funding to persue this study in UK in the form of University Postgraduate Studentship;
- the Jamia Millia Islamia, New Delhi for granting me study leave to persue this study in UK;
- the academic and technical staff of the school for all their help during my study; especially I would like to thank Mr. Chris Burnside for sharing his knowledge of computer hardware and software systems.
- my parents, brothers and sisters for providing me a wonderful family;
- Salma for giving me cheerful company throught the course of this work and bearing with all the difficulties of bringing up our children and running the home smilingly;
- Saima for filling my evenings with happy moments after a days work;
- Asim for all the motivation he gave me in completing this work within the stipulated period;
- all my friends who made our stay in Edinburgh an unforgettable and rewarding experience.

Contents

List of figures

List of tables

Acronyms and abbreviations

ARIMA	auto-regressive-integrated-moving-average
BCM	billion cubic metre
CCVP	California Central Valley Project
DP	dynamic programming
DDP	differential dynamic programming
DDDP	discrete differential dynamic programming
DPSA	dynamic programming successive approximation
DWD	directorate of water development
EA	evolutionary algorithm
EC	evolutionary computation
EDS	existing development situation
ELQG	extended linear quadratic Gaussian
FDS	future development situation
FWL	flood water level
G.	Gunung (volcano)
GA	genetic algorithm
HEC	hydrologic engineering center
HWL	high water level
IDP	incremental dynamic programming
K.	Kali (river)
KW	killo watt
LP	linear programming
LQG	linear quadratic Gaussian
LWL	low water level
MAM-DP	multilevel approximate dynamic programming
MCM	million cubic metre
MES	Microsoft Excel solver
MW	mega watts
NBS	net basin supply
NLP	non-linear programming
NN	neural network

OCT	optimal control theory
PC	personal computer
PFA	power forecasting approach
POA	progressive optimality algorithm
QAP	available Perning discharge
QP	quadratic programming
RC	rule curve
SA	simulated annealing
SDP	stochastic dynamic programming
SLP	successive linear programming
SMGA	structured messy genetic algorithm
SQP	sequential quadratic programming

1. INTRODUCTION

1.1 Introduction

Water is the most essential resource of all life on Earth. The spatial and temporal distribution of water is highly variable and is dependent upon the climatic factors that are beyond human control. Due to tremendous population growth and extensive industrial and agricultural development, the demand on water resources is increasing everywhere in the world. In some parts, the characteristics of water supply and demand pose few problems for agricultural, domestic, and industrial users. In other areas, including much of the developing world, physical, social and political factors make effective water resource management vital.

The situation is critical in developing countries as the gap between water demand and supply has been continuously widening. This has led to an increased emphasis on the optimal management of the available resources. Rigorous planning and management of water resources is required for long term sustainable resource development. The need for optimal management of existing water resource systems as well as the optimal development of the new ones is now universally acknowledged. Water resource systems are an important part of the infra-structure of every country, particularly the developing ones. In addition to the basic purpose of supporting life, they serve a multitude of water uses such as water supply, hydropower generation, recreation, irrigation, flood control, navigation and wild life maintenance.

Optimising the economic benefits of water resource systems is a classical and persistent problem. The solution to the problem is difficult because of the large number of variables involved, the non-linearity of system dynamics, the stochastic nature of future inflows, and other uncertainties of the system. Nevertheless, a number of mathematical programming techniques have been developed to aid derivation of optimal operating strategies for water resource systems. Most of these techniques perform satisfactorily for the problems they are developed for. A generic methodology that can handle problems in their general form has not yet been identified however. For this reason, there is a continuing need to improve and extend existing optimisation techniques as well as to explore new ones.

The rest of this chapter is organised as follows. Section 1.2 presents a brief survey of the current state of research in the area of reservoir systems optimisation. Following this, section 1.3 gives an exposition of the principal motivation behind the research undertaken in this thesis. Section 1.4 gives an outline of the objectives of this research. The contributions made in this thesis are outlined in section 1.5. At the end, section 1.6 presents the thesis layout.

1.2 Background

The planning and operation of reservoir systems for sometimes conflicting purposes can be a complex process. This is primarily because the potential benefits and risks associated with the operation of reservoir systems can be large. Maximum benefits, which are usually defined in some economic terms, need to be achieved without reducing the reliability of the system, thus making the whole process even more complicated. Risk is inherent in all reservoir operations, be it risk of emptying or the risk associated with flood releases, and require careful attention by reservoir managers. Because of the complexities of multi reservoir systems, optimisation models are often used to determine optimal operating policies. Simulation models have also been widely used for this purpose. Optimisation models can be used in planning to determine the optimum size of reservoir to construct, or in operations to determine optimal release policies.

During the last three decades, one of the most important advances made in the field of water resources engineering has been the development of optimisation techniques for planning, design and management of complex water resource systems. The recent rapid increase in computer technology has made the development of sophisticated mathematical models for the analysis of water resource systems possible. These models are increasingly being used by system managers to determine decision alternatives which are optimal in some defined sense. The optimisation of reservoir systems operation usually involves the search through large decision spaces for optimal parameter sets. Often, the decision space is too large for a complete search. This has motivated the development of various optimisation procedures. However, despite the extensive research carried out in the last three decades, reservoir control still remains an active research field.

Most optimisation procedures are based upon some type of mathematical programming technique such as linear programming (LP), non-linear programming (NLP), or dynamic programming (DP). DP (Bellman 1957; Bellman and Dreyfus 1962) has long been recognised as a powerful approach in the analysis of water resource systems. The major reason DP is so attractive is that it can handle non-convex, non-linear and discontinuous objective functions without difficulty. Constraints on both decision and state variables introduce no difficulties. Hall and Buras (1961) were the first to propose the application of DP to determine the optimal returns from the reservoir systems. Young (1967) developed optimal operating rules for a single reservoir using DP. Since then, DP has been the most widely used technique for reservoir systems operation. The technique, however, becomes computationally bounded for large systems due to the "curse of dimensionality", a term used for huge memory and computational time requirements of a multidimensional problem (Bellman 1957).

Many forms of DP including decomposition, aggregation, and successive approximation have been developed to alleviate the problem of dimensionality with varying degrees of success. The most regarded and widely used variant of DP is the discrete differential dynamic programming (DDDP). The technique was first proposed by Larson (1968) under the nomenclature of incremental dynamic programming (IDP). Another approach that overcomes the dimensionality problem is the DP successive approximation (DPSA) technique (Larson 1968; Trott and Yeh 1973). Murray and Yakowitz (1979) have developed differential DP (DDP) for multi reservoir control problems. Turgeon (1981a) has reported the use of progressive optimality algorithm (POA), which can handle large scale systems. The implementation of DDP and the POA is more complicated than that of DDDP and DPSA. Although DDP appears free from dimensionality problems, it requires that the objective function be differentiable and that the constraints are linear.

Approaches based on optimal control theory (Pontryagin et al. 1962) (OCT) have also been used for reservoir systems optimisation. These approaches do not suffer from dimensionality problems but require that the objective function should be differentiable. OCT was originally developed for unconstrained problems. To handle the constraints of the problem, a penalty function approach is required. Wasimi and Kitanidis (1983), Papageorgiou (1985), Georgakakos and Marks (1987), Georgakakos (1989), McLaughlin and Velasco (1990), and

Mizyed et al. (1992) have all reported the use of the approach. Many other optimisation techniques have been developed with their own merits and demerits.

1.3 Motivation for Work

This section outlines the need for undertaking this research. The increasing gap between the demand and supply of water in most parts of the world has made it imperative to optimise the utilisation of available water resources. As water resource systems have grown larger and more complex, the importance of optimal operation and planning of these systems has increased. The investment costs and operating expenses are often so large that even small improvements in system utilisation can result in substantial financial gains. Although the existing systems are being managed with some type of optimisation techniques, there is potential for improvement in the operation schemes.

The usefulness of DP for multi reservoir systems is limited by the huge demand it induces on computational resources. With the conventional DP procedure it is usually not possible to consider the simultaneous operations of more than two reservoirs due to the curse of dimensionality (Yakowitz 1982). Discretization of state space is required to solve the reservoir operation problems by DP and this is the major reason for the problem of dimensionality. Recently, the emphasis has shifted from DP based methods to those which do not require discretization of state space. DDP (Murray and Yakowitz 1979) and OCT approaches (Wasimi and Kitanidis 1983; Georgakakos and Marks 1987; Georgakakos 1989) are among such methods. Many successful applications of these techniques have been made in reservoir operation and planning studies but no general algorithm exists.

Many optimisation algorithms have been developed to obtain approximate solutions to complex operating and planning problems associated with the reservoir systems. Most of these algorithms have their own limitations. Some require discretization of the state and decision space while others require that the objective function is differentiable. A major drawback of most DP based iterative methods, such as DDDP and DPSA, is that they may converge to a local optimum, or may not converge at all, unless the problem itself satisfies some stringent conditions. Solution of multi reservoir systems with high dimensionality and complex objectives still poses a challenge to researchers and no generalised technique is available which can fully solve problems of all types. For this reason, there arises a need for research into robust and more efficient algorithms to overcome the problem of dimensionality and the limitations associated with existing techniques. This has been the focus of much recent research (Papageorgiou 1985; Hiew 1987).

To address the above, optimisation based on the genetic algorithm (GA) approach is investigated in this thesis. The development of GA procedures for reservoir control problems is largely motivated by the dimensionality problems associated with the conventional DP approach. GAs are a class of search algorithms which have considerable potential for employment in the planning and analysis of water resource systems. GAs are theoretically and empirically proven to provide a robust search in complex spaces (Goldberg 1989). Many works (DeJong 1975; Michalewicz 1992) have established the validity of the GA technique in function optimisation. A growing number of researchers are showing an interest in the field of GAs due to the relative simplicity of use of these algorithms. Although the algorithm is simplistic in nature, it is powerful. GAs can easily accommodate discontinuous, non-differentiable and multimodal functions, and it is in this area that the strength of GAs lie.

GAs are a class of combinatorial optimisation methods that searches for solutions of complex problems using the mechanics of natural selection. They use a stochastic search procedure inspired by biological evolution to obtain better solutions to survive and propagate to successive generations. The execution time for GAs increase at a significantly lower rate compared to traditional methods like DP (Goldberg 1989; Esat and Hall 1994). As the complexity of a problem increases, GAs are expected to be computationally efficient. The use of GAs is increasing at a rapid rate for solving problems that have been found difficult to solve by traditional optimisation techniques. Davis (1991), Michalewicz (1992), and Dasgupta and Michalewicz (1997) have reported many successful applications of GA to real-world problems from a wide spectrum of fields. GAs have had a relatively slow uptake in water resources systems optimisation compared to other disciplines. The literature describing the application of the GA technique to the operation of reservoir systems is not extensive.

1.4 Thesis Objectives

The main objective of this research is to develop and evaluate the GA approach for the operation of multi reservoir systems. Specific objectives are as follows:

- to survey the use of optimisation techniques for the operation of multi reservoir systems;
- to develop and apply the GA approach to problems with known solutions and compare the performance with other techniques;
- to apply the developed approach to real world problems with deterministic and stochastic inflows;
- based upon the above studies, identify the merits and the limitations of the approach.

The first objective is achieved through review of literature focusing on the use of optimisation models in the operation of large scale systems, and is presented in chapter 2 of this thesis. Particular attention has been given to techniques that aim to alleviate the dimensionality problem posed by DP. The second objective is achieved through the application of the GA technique to a four reservoir problem formulated by Larson (1968). Further verification of the GA model is carried out by applying the approach to a modified four reservoir problem, and to a ten reservoir problem introduced to the literature by Murray and Yakowitz (1979). These problems were chosen because they offered an opportunity to test the performance of GA against the known global optima.

The practicality of the developed GA approach is evaluated through application to a reservoir system in Indonesia. Finally, real time operation of the Equatorial Lakes system on the River Nile in Africa is considered using the GA approach. Based on the results of these applications, the merits and limitations of the GA approach have been identified and discussed, thus constituting the last objective of this research.

1.5 Thesis Contributions

This section outlines some of the major contributions of the research presented in this thesis. A generic methodology based on the GA approach has been developed for optimisation of multi reservoir systems. The approach developed here does not require discretization of state variables and is equally applicable to problems with discontinuous and non-differentiable objective functions. The GA approach leads to a computational procedure that has memory and computational requirements significantly less than those of conventional procedures for large reservoir systems. These computational and memory savings are obtained without any loss in the generality of the problems that can be solved.

The well known four reservoir problem is solved first and the solution has been compared to that obtained in the past by different researchers. Evaluation of alternative GA formulations have been carried out, and sensitivity of GA performance to different parameters is also analysed. A non-linear four reservoir problem has been solved, along with a problem with extended time horizons. A ten reservoir problem has also been solved satisfactorily. For the non-linear four reservoir problem, the optimum was obtained thus indicating that a GA can indeed reproduce an optimum.

After the GA approach has been verified, it is applied to a case study of a practical reservoir system in Indonesia. In this study, a deterministic finite horizon reservoir operation problem has been solved. Optimal operating rules have been derived for the existing development situation in the basin, and also for two future water resource development scenarios using the critical period hydrology. Subsequently, the methodology for real time operation of reservoir systems is developed and applied to the Equatorial Lakes system of the River Nile in Africa. In all the cases considered, the GA results have been compared to those obtained by other techniques, and have been found to be satisfactory. DDDP models have also been developed for the four reservoir problem, the Brantas Basin and the Equatorial Lakes system. Simulation models have been used to evaluate the economic performance of the Brantas Basin system and the Equatorial Lakes system for a number of operating criterion.

GAs have some distinct practical advantages over traditional optimisation techniques. With GA, discretization of storage space is not required thereby eliminating one of the major causes of dimensionality problem associated with DP. A GA requires only a numerical measure of the solution, and therefore can easily handle discontinuous and indifferentiable objective functions. The computational complexity for GA grows at a significantly slower rate than for the DP. Savings in terms of execution time and memory requirements are realised when compared to DP and to its variants such as DDDP for large systems. GAs generate a number of alternative solutions close to the optimum which gives added flexibility to the operator of a complex reservoir system.

1.6 Thesis Organisation

The rest of the thesis is organised in 8 Chapters and three appendices.

Chapter 2 provides a review of the optimisation models used in reservoir operation studies. The basic concepts of commonly used mathematical programming techniques for reservoir systems optimisation are also explained in this chapter.

Chapter 3 presents the theory and fundamentals of the GA approach. The procedure for problem formulation, different representation schemes, and reproduction operators are also described. Subsequently, a brief review of the application of GA to water resources problem in general and reservoir operations in particular is presented.

Chapter 4 describes the application of the GA approach to the well known four reservoir problem. Several alternative formulations of GA have been evaluated and sensitivity of GA performance to various parameters has been analysed. The application of GA to a non-linear four reservoir problem and to a ten reservoir problem is also described in Chapter 4. A four reservoir problem with extended time horizon is also considered.

Chapter 5 describes the Brantas Basin located in Indonesia and also gives the details of a simulation model of the basin. The application of GA in the determination of optimal reservoir operating rules for the Brantas Basin is presented in chapter 6. In chapter 7, the methodology for real time operation of multi reservoir systems is developed and demonstrated through application to the Equatorial Lakes system on the River Nile in Africa.

Chapter 8 provides the conclusions of the research. The chapter lists the achievement of the research and outlines the limitations of the work carried out. Recommendations for further research in the application of GAs to water resource systems are also provided.

2. LITERATURE REVIEW

2.1 Introduction

This thesis describes the development and application of genetic algorithms (GAs) for optimisation of multi reservoir systems. The optimal operation of multi reservoir systems is a subject of great practical and economic significance in the field of water resources engineering. This is reflected in the abundance of literature in the field. The continuing research effort is sustained because the outputs generated by reservoir systems have an influential social, economic, and environmental impact. In order to establish the context and the need of the research undertaken clearly and coherently, it is necessary to discuss various optimisation techniques that have been used in the past by different researchers. This chapter discusses these techniques and presents a review of optimisation models used in reservoir operation problems.

Following this introduction, section 2.2 describes mathematical programming techniques commonly used in reservoir systems optimisation. A review of applications of these techniques to multi reservoir operation problems is presented in the same section. Applications of different techniques to real time operations is presented in section 2.3. Non-conventional optimisation approaches used in engineering applications are briefly discussed in section 2.4. Finally, section 2.5 provides the concluding remarks.

2.2 Mathematical Programming Techniques

Most optimisation models are based on some type of mathematical programming technique. Many successful applications of these techniques to reservoir operation studies have been reported in the literature, but no universally proven technique exists. Excellent treatment of these techniques can be found in the works by Loucks et al. (1981) and Mays and Tung (1992). A survey of dynamic programming (DP) models applied to water resources planning problems was presented by Yakowitz (1982). The application of various optimisation models to reservoir operation problems has been reviewed by Simonovic (1992) and Wurbs (1993). Yeh (1985) provides an excellent state-of-the-art review of reservoir management and operation models. According to Yeh (1985), the techniques being commonly used by the researchers can be broadly classified as follows:

1. Linear programming (LP)
2. Non-linear programming (NLP)
3. Dynamic programming (DP) including, discrete differential DP (DDDP), differential DP (DDP), successive approximation DP (SADP), and stochastic DP (SDP)
4. Simulation

2.2.1 Linear Programming

In a problem where all the objective and constraint functions are linear, LP can be used in the optimisation of reservoir systems. It has been one of the most widely used techniques in water resources management due to its simplicity and adaptability.

A typical LP model is

$$\text{Minimise or Maximise} \quad \mathbf{Z} = \mathbf{C}^{\mathbf{T}}\mathbf{X} \tag{2.1}$$

$$\text{subject to} \quad \mathbf{AX} \geq \mathbf{b} \tag{2.2}$$

where $\mathbf{X} \geq \mathbf{0}$ is a n dimensional vector of decision variables, **C** is a n dimensional vector of objective function coefficients; **b** is a m dimensional vector of right hand side of (2.2); **A** is a $m \times n$ matrix of constraint coefficients; and **T** represents the matrix transpose operation.

Dorfman (1962) demonstrated the application of LP to a water resource problem with three versions of a model, each with increasing complexity. The objective was to maximise the economic benefits of water use while satisfying the constraints of the problem. In all the three versions, both storage capacities and target releases were treated as decision variables. The first version of model involved a simple LP application to a simplified river basin planning problem. In the second version, critical period hydrology was used. In the third version, the model treats inflows stochastically.

Shane and Gilbert (1982) and Gilbert and Shane (1982) describe a model called HYDROSIM used to simulate the Tennessee Valley Authority reservoir system based on established operating strategies. The model employed LP to compute reservoir storages, and hydropower generation for each period of operating horizon. Palmer and Holmes (1988) incorporated a LP model in the Seattle Water Department integrated drought-management expert system. The model was used to determine optimal operating policies and system yield based upon the objectives of maximising the yield, and minimising the economic losses associated with deficits from a specified target.

Randall et al. (1990) used LP to study the operation, during drought, of a water resource system consisting of multiple reservoirs, groundwater, treatment plants, and distribution facilities. The objectives was to maximise the net revenues, which were the difference between the cost of production and the selling price of water; maximise reliability, expressed as the minimum of the ratios of consumption to demand for each water use deficit; maximise reservoir storage at the end of the operating horizon; and maximise the minimum flows in the streams. Crawley and Dandy (1993) used LP to develop a planning and operational model for the Adelaide headworks system in South Australia. The objective was to determine optimal sequences of pumping and transfers for the system so as to minimise pumping cost while maintaining a satisfactory level of reliability within the system. Martin (1995) describe an optimisation procedure based on LP to maximise the power generation over a 24-hr period from the Highland Lakes of Lower Colorado river in Texas.

The application of LP requires the linearisation of constraints and of the objective function, which for most of the practical reservoir systems are non-linear functions. This limits the application of LP to problems with linear functions. Non-linear functions can be approximated by linear functions, and successive LP (SLP) can be used to approximate the solutions. Grygier and Stedinger (1985) and Hiew (1987) describe the application of SLP, among many other techniques, to multi reservoir optimisation problems. Reznicek and Simonovic (1990) describes the application of SLP to Manitoba Hydro system in Canada. The objective was to maximise the power production from the system. Since the power was not linearly related to the release, Taylors series expansion was used to linearise the power function. Such simplification may, however, lead to reduction in the value of the optimisation results.

2.2.2 Non-linear Programming

NLP technique can be applied where either the objective function or constraints are non-linear. NLP can effectively handle a non-separable objective function and non-linear constraints. A general NLP problem can be expressed in the form

Minimise $F = f(x_1, x_2, \ldots, x_n)$ *(2.3)*

subject to $g_i(x) = 0$ $i = 1, m$ *(2.4)*

where $\underline{x}_j \leq x_j \leq \overline{x}_j$ $j = 1, n$ *(2.5)*

in which F is to be minimised subject to m constraints expressed by function $g(x)$, n is the number of decision variables, and (2.5) is a bound constraint for the jth decision variable x_j with $\underline{x}_j$ and $\overline{x}_j$ being the lower and upper bounds, respectively.

NLP has not been very popular due to the computational complexity of the approach for multi reservoir systems optimisation. Lee and Waziruddin (1970) applied NLP to a theoretical system of three reservoirs in series with the objective of maximising a non-linear function of irrigation releases and storages in the reservoirs. Applications of NLP have also been reported by Simonovic and Marino (1980), Rosenthal (1981), and Guibert et al. (1990).

A special case of NLP is quadratic programming (QP) where the degree of various terms in the objective function is 2, 1, or 0. A quadratic optimisation model for the California Central Valley Project (CCVP) has been presented by Marino and Loaiciga (1985). The model was compared with an LP model, and it was found that a significant increase in the total energy production could be obtained using the QP model. Diaz and Fontane (1989) applied sequential QP (SQP) to determine optimal economic returns from hydropower generation for a multi reservoir system in Argentina. The SQP approach was found to be superior to SLP in terms of the execution time and the value of the objection function achieved. Wardlaw et al. (1997) used QP to solve water allocation problem to the Lower Ayung irrigation system on the island of Bali in Indonesia. The objective was to maximise crop production while maintaining equity in water supply between irrigation schemes and the irrigation blocks within the schemes.

A large number of NLP software packages are commercially available. Most commonly used packages include GAMS, INRIA, LINDO, LINGO, Mathworks, NAG and OSL. For multiple reservoir systems, the number of constraints is large because they deal with similar subsystems repeated in time or location. Therefore, NLP requires large amount of storage and execution time when compared to other methods limiting its applicability to large systems (Yeh 1985). The use of NLP is further limited to problems that are smooth and continuous because it requires the calculation of derivatives for its search procedure.

2.2.3 Dynamic Programming

DP (Bellman 1957) is the most commonly used method for the optimisation of reservoir systems as these are characterised by large number of non-linear and stochastic features that can be translated into a DP formulation. DP is an enumeration procedure used to determine the combinations of decisions that optimises overall system effectiveness as measured by a criterion function. It is capable of treating non-convex, non-linear and discontinuous objective and constraint functions, and this is the greatest advantage of DP. Constraints on both decision and state variables introduce no difficulties. In fact, the constraints speed up the computational procedure.

The key feature of DP application is that it is usually identified as serially or progressively directed for an operational or planning problems, respectively. The operation of reservoirs is a multistage decision process and DP is particularly suited to such problems. The problem is divided into stages with a decision required at each stage. The stages usually represent different points in time and each stage should have a finite number of states associated with it. In reservoir operation studies, the state usually represents the amount of water in the reservoir at a given stage. If DP is used for determination of reservoir releases, these form the decision variables. The stage to stage transformation is carried out by the continuity equation subject to constraints on storages and releases. The recursive equation of DP can be written as

$$F_n(s_n) = \max\left[V_n(s_n, d_n) + F_{n-1}(s_{n-1})\right] \qquad (2.6)$$

where s_n is the state variable, d_n is the decision variable, $V_n(s_n, d_n)$ is the objective function value, $F_n(s_n)$ is the cumulative return at stage n with $F_0(s_0)$ known, and $s_{n+1} = g(s_n, d_n)$ is the stage to stage transformation function.

Given a starting state, the DP model calculates the vector of states reached in the next stage for all possible combinations of states. This require exhaustive searching over all possible states in the next stage because the next state can lie anywhere in the space of admissible states. The decision variables may not be discretized in a DP algorithm. Rather than using the state in the current stage and a decision set to calculate the states in the next stage, the decisions that would be required for the states in the current stage to go forward to feasible states in the next stage are computed. This eliminates the need for interpolation and results in considerable reduction in computing time. For each state reached in the next stage, the cumulative return based upon the objective function is calculated and the value assigned to the new state reached. As the same state can often be reached from several different preceding states, each having a different return, only the maximum return is stored with the preceding state also being recorded. The model then moves to the next stage and repeats the process for all states with a non-zero return. When the final stage is reached the model traces back through each stage to find the sequence of states and releases that generated the maximum return at each stage.

DP has been successfully used by many researchers for optimisation of water resource systems. Hall and Buras (1961) were the first to propose the application of DP to determine optimal returns from reservoir systems. Young (1967) developed optimal operating rules for a single reservoir using DP. A series of synthetically generated inflow sequences were used to derive a set of optimal release trajectories. These trajectories were then related to various system variables through regression analysis. As a result, an operating rule expressed as a function of a set of system variables was obtained. Allen and Bridgeman (1986) used DP for optimal scheduling of hydroelectric power. Applications of DP to reservoir operation problems have also been reported by Opricovic and Djordjevic (1976) and Collins (1977). Extensive review of DP applications to reservoir systems can be found in the works by Yakowitz (1982) and Yeh (1985).

The disadvantages associated with DP are the huge requirements in terms of computer memory and execution time. The approach breaks down on the problems of moderate size and complexity, suffering from a malady labelled the 'curse of dimensionality' by its creator Bellman (1957). For a system with n state variables and k levels of discretization, there exists k^n combinations that need to be evaluated at each stage of analysis. The usefulness of DP when applied to multiple reservoir systems is therefore limited by the "curse of dimensionality" which is a strong function of the number of state variables and the levels of discretization used. The application of DP to problems with more than two or three state variables still remains a challenging task on present day computers.

A traditional and simplistic procedure for reducing computational effort in DP is the iterative coarse grid method. The problem is first solved using a coarser discretization of the state variables. Based upon the resulting solution, revised bounds on the state variables are defined and the grid size is then reduced. The iterative procedure is repeated until the grid size has been reduced to a desired precision. The algorithm is stopped when no further improvement in the value of the objective function can be obtained. This procedure, however, cannot guarantee a global optimum. Besides, it also does not resolve the dimensionality problem.

To overcome the dimensionality problem imposed by DP to some extent, use of LP in combination with DP has been reported by many researchers. In a combined LP-DP procedure, the stage to stage optimisation is carried out by LP, and DP is used for determining optimal policy over a number of stages. Becker and Yeh (1974) applied a combined LP-DP approach to optimal real time operations associated with CCVP. The LP minimised the loss in potential energy of the stored water in the reservoirs resulting from any release policy in each period. The multiperiod optimisation was carried out by embedding the LP solutions in a deterministic forward DP. The LP-DP combination has also been used by Takeuchi and Moreau (1974), Becker et al. (1976), Yeh et al. (1979), Yeh and Becker (1982), and Marino and Mohammadi (1983). The non-linearities are handled using an iterative technique, such as SLP. Grygier and Stedinger (1985) describe the application of SLP, an optimal control algorithm (Pontryagin et al. 1962), and a combined LP-DP algorithm to a multi reservoir system. The optimal control algorithm is based on Pontryagin's maximum principle (Pontryagin et al. 1962) and involves the solution of Kuhn-Tucker necessary conditions of optimality. The optimal control algorithm for the problem solved by Grygier and Stedinger (1985) executed five times faster than the SLP. The LP-DP algorithm took longer to execute and produced comparatively inferior solutions.

2.2.4 Discrete Differential Dynamic Programming

Many variants of DP have been developed over time to alleviate the problems of dimensionality. Notable among these are incremental dynamic programming (IDP), DPSA and DDDP. These are iterative techniques and start with the assumption of a trial trajectory. DDDP is specifically designed to overcome the dimensionality problem posed by DP. The technique uses the same recursive equation as DP to search among the discrete states in the state-stage domain. Instead of searching over the entire state-stage domain for the optimum, the optimisation is constrained only to a part of the state-stage domain saving computer time and memory. The method starts with the selection of a trial state trajectory satisfying the boundary conditions. Several states, located in the neighbourhood of a trial trajectory can be introduced to form a band called a corridor around the trial trajectory. The traditional DP approach is applied to optimise within the defined corridor. Consequently, an improved trajectory is obtained which is adopted as the new trajectory to form a new corridor. This process of corridor formation, optimisation with respect to the states within the corridor and trace back to obtain an improved trajectory for the system is called an iteration. The procedure is repeated for a number of iterations until no further improvement in the value of the objective function can be obtained.

Larson (1968) obtained the solution to the four reservoir problem using IDP. Hall et al. (1969) used a different version of IDP and solved a two reservoir system. The major difference between the two versions is that the time interval used in the computation is variable in the former and fixed in the latter. Another version called DDDP was developed by Heidari et al. (1971) which could be seen as a generalisation of IDP. They solved the four reservoir problem formulated by Larson (1968). The terms IDP and DDDP have been used interchangeably in water resources applications. In the standard IDP of Larson (1968), the number of discretizations are limited to three per state variable. The computational burden and memory requirements for such an approach is a function 3^n, where n is the number of state variables. IDP does overcome the dimensionality problem to a large extent but requires stringent conditions to be satisfied for implementing the procedure. The IDP method require that there must exist a function $g_t(s_t, s_{t+1})$ such that for every pair of states s_t and s_{t+1},

$$g_t(s_t, s_{t+1}) = r_t \qquad (2.7)$$

where r_t is the release in time step *t* such that $s_{t+1} = f_t(s_t, r_t)$ *(2.8)*

In IDP, an optimal solution can only be obtained if all the states in the corridor are accessible from one stage to another. When the optimisation is restricted to three states in each stage, the above condition may not be satisfied and IDP may converge to a non-optimal solution. When the optimisation is carried out over the entire decision space, the method becomes time consuming and memory requirements are also high but the likelihood of locating the optimum is increased. Turgeon (1982) demonstrated that IDP may lead to non optimal solutions. Suggestions were then made to adjust the increment sizes in each stage to obtain the desired results. The choice of initial trial trajectory is vital for good convergence in all iterative algorithms. For complex systems, the determination of feasible trial state trajectories is not a trivial task.

Another approach which overcomes the dimensionality problem is the DP successive approximation technique (DPSA). The technique was first proposed by Larson (1968). Trott and Yeh (1973) used successive approximation technique in combination with IDP. The original multiple state variable DP problem was decomposed in a series of subproblems of one state variable. To demonstrate the technique, a six reservoir problem was solved. The optimisations were carried out with respect to a single reservoir while keeping the states in the other reservoirs fixed. The procedure is repeated for other reservoirs until the convergence criteria is satisfied. Successive approximation is one-at-a-time optimisation technique and its common drawback is convergence to a local optimum. Extension to successive approximation technique has been reported by Nopmongcol and Askew (1976), who suggested higher level combinations, such as two or three-at-a-time combinations. The technique was demonstrated through application to the four reservoir problem.

The requirement that the state variables be discretized is a major cause of computational complexity. Jacobson and Mayne (1970) developed differential DP (DDP) for multi state problems. Murray and Yakowitz (1979) applied DDP to multi reservoir control problems. The important feature of the technique is that discretization of state and decision space is not required. The method requires a quadratic approximation of the objective function. Application of the method to a ten reservoir problem by Murray and Yakowitz (1979) has demonstrated its effectiveness. Murray and Yakowitz (1979) also solved the four reservoir problem using DDP and showed that the global optimum was obtained in much lesser number of iterations than in DDDP. Application of DDP to estuary management has been reported by Li and Mays (1995). However, the technique requires that the objective function is differentiable and that constraints are linear.

Howson and Sancho (1975) developed a progressive optimality algorithm (POA) for multistate problems. Turgeon (1981b) applied the algorithm to an example system consisting of four reservoirs in series. The most promising aspect of the approach is that the state variables do not have to be discretized. The algorithm, however, required 823 iterations to converge to an optimal solution for the simple example considered. The approach has merit in that it overcomes the dimensionality problem. A binary state DP algorithm has been developed by Ozden (1984) in which the optimisation is constrained within a subset consisting of only two states at each stage. One of the states is the component of the optimal trajectory found in the previous iteration. The second value is defined according to the shifting direction of the optimal trajectory in the state space at previous iteration. This procedure differs from DDDP in the manner in which the states are chosen for next iteration.

2.2.5 Optimal Control Theory

Optimal control theory (OCT) also provides an efficient means of overcoming the dimensionality problem. State and release variables need not be discretized in such approaches. The requirement that the objective function must be differentiable limits the application of approach to some extent. Storage constraints are not automatically satisfied but can be handled by use of appropriate penalty functions. Wasimi and Kitanidis (1983) used linear quadratic Gaussian (LQG) approach based on OCT to solve a multi reservoir control problem. They used a set of differential equations to describe the dynamics of a reservoir system and employed a quadratic penalty function to keep the state variables within bounds. Papageorgiou (1985) solved a ten reservoir problem using the same approach. Georgakakos and Marks (1987) modified the LQG approach to include the non-linear dynamics and applied it to the operation of High Aswan dam. Further extensions to the approach were made by Georgakakos (1989) to handle non-Gaussian features that frequently characterise reservoir inflows. An LQG model has also been developed by McLaughlin and Velasco (1990) for stochastic monthly operation of a two reservoir system. Hooper et al. (1992) describe application of ELQG to Salt River project, Arizona. The use of ELQG to reservoir operation problems has also been reported by Georgakakos et al.

(1997). Zhao and Mays (1995) describe an estuary management model based on stochastic LQG control.

2.2.6 Simulation

Simulation is a modelling technique used to approximate the behaviour of a system on a computer, representing all the characteristics of the system largely by a mathematical or algebraic description. Simulation models provide the response of the system to certain inputs, which include decision rules that allow the decision makers to test the performance of existing systems or a new system without actually building it. A typical simulation model for a water resources system is simply a model that simulates the interval-by-interval operation of the system with specified inflows at all locations during each interval, specified system characteristics and specified operating rules.

Optimisation models aim to identify optimum decisions for system operation that maximises certain given objectives while satisfying the system constraints. On the other hand, simulation models are used to explore only a finite number of decision alternatives so that the optimum solution may not necessarily be achieved. However, many simulation models now involve a certain degree of optimisation and the difference between the optimisation and simulation models is becoming less distinct. For a given operating criteria, the performance of a reservoir system may be evaluated by analysing the computed time sequence of levels, storage, discharges, hydropower etc. The procedure can be repeated for a number of inflow sequences to arrive at a statistical measure of the system.

Simulation models have been routinely applied for many years by water resource development agencies. Yeh (1985) and Wurb (1993) present reviews of a number of such models. An excellent treatment of the subject of computer simulation in hydrology has been provided by Fleming (1975). A total of 19 simulation models have been presented in the text. The background and structure of each model is discussed, and the functions used to represent the major processes involved are described. The input/output requirements and the range of application of models is also discussed. Simulation models have also been extensively used in combination with optimisation models. Karamouz and Houck (1982) describe an algorithm that cycles through an optimisation model, a regression analysis and a simulation model to develop reservoir operating rules. Labadie et al. (1987) presents an application of a simulation model in estimating the reliable power capacity of a reservoir system.

Recently, researchers have tried to incorporate optimisation methods within the simulation models. Wardlaw (1993) has developed a simulation model which incorporates economic functions for hydropower, agriculture and fisheries production. The model was used to assess the economics of alternative strategies for water resources development in the Brantas Basin in Indonesia. Jain et al. (1996) describe the application of a simulation model to reservoir operation studies of Sabarmati system in India. Operating procedures were derived for all the four reservoirs of the system.

2.3 Applications to Real Time Operation

This section presents a review of optimisation models applied to real time operations in particular. Real time operation of reservoir systems can be a formidable task due to the large scale of some systems, and the non-linearity and complexity of the objective functions often encountered in practice. The literature describing real time reservoir operations is vast. It is astonishing to note that the first application of stochastic reservoir operation model was reported by Little (1955). Since then, a large number of applications have been reported in the literature. Yakowitz (1982) remarked that two reservoir systems are the largest to be solved by stochastic DP. On the other hand, applications of variants of DP to deterministic problems have been reported for systems of upto 10 reservoirs, for example, by Murray and Yakowitz (1979). This is not totally unexpected as the dimensionality problem is much more severe for stochastic problems than for the deterministic ones. However, problems with larger dimensions have been tackled by Turgeon (1981a; 1981b), and Archibald (1997) using variants of DP.

SDP was used by Louks and Falkson (1970), Dudley and Burt (1973), Torabi and Mobasheri (1973), Mawer and Thorn (1974), and Stedinger et al. (1984). Lee et al. (1992) describes the application of a SDP model to Lake Shelbyville in Illinois. Becker and Yeh (1974) used DP in combination with LP for real time operation of multiple reservoir systems. Turgeon (1980) describes two variations of DP to a system of reservoirs all in parallel using stochastic inflows. In the first version, the original problem was divided into a series of problems with a single state variable. The solution to such a problem leads to an optimal local policy for each reservoir. In the second version, the original *n* state problem is decomposed in *n* subproblems of two state variables resulting in a suboptimal operating policy for the whole system. Turgeon (1981b) describes another decomposition method for a system of reservoirs in series. The original *n* state problem is decomposed in *n*-1 problems of two state variables which are solved by DP. Archibald et al. (1997) describe an aggregate SDP model of multi reservoir systems. The method uses a three dimensional representation of the system which consists of a detailed model of the focus reservoir, an approximate model of reservoirs whose releases can reach that reservoir, and an approximate model of the remainder of the system. SDP is then used to solve the subproblems routinely.

Turgeon's algorithms were developed for a system of reservoirs either all in parallel (Turgeon 1980) or all in series (Turgeon 1981b). Poonambalam and Adams (1996) developed a technique, called the multilevel approximate DP (MAM-DP), to accommodate any type of reservoir system configuration. In Turgeon's algorithms two state variables were used to represent the reservoir system while in MAM-DP the number of state variables to be used depend upon the size of the problem and the computing resources available.

DDDP has also been used with stochastic inflows. Application of DDP to a problem with stochastic inflows has been described by Trezos and Yeh (1987). They employed a solution procedure similar to that of Murray and Yakowitz (1979) but used stochastic inflows rather than deterministic inflows. Kuo et al. (1990) reported application of stochastic DDDP to a two reservoir system in the Tanshui River Basin, China For the forecast sequence of streamflows, an initial feasible release policy was determined using a simulation model. DDDP is then used to improve the policy determined from the simulation model.

Marino and Loaiciga (1985a) presented a methodology for obtaining optimal reservoir operation policy for a large scale reservoir system using a sequential dynamic decomposition algorithm. Marino and Loaiciga (1985b) also describe a QP model for the CCVP. The results of QP showed an increase in energy generation over that achieved using LP. Karamouz et al. (1992) describe a three-step procedure for deriving operating rules for a multi reservoir system. The optimisation model is used to generate optimal solutions for different synthetically generated sequences. The solutions are analysed in a regression procedure to obtain a set of operating rules. A simulation model is then used to evaluate the performance of these rules. Similar procedures have also been reported by Bhaskar and Whitlach (1980), and Karamouz and Houcks (1982).

2.4 Other Optimisation Approaches

A number of other optimisation approaches are in use to solve difficult engineering problems. These include simulated annealing (SA), neural networks (NNs) and evolutionary computation (EC). All such approaches are based upon the mechanics of natural processes.

2.4.1 Simulated Annealing

Simulated annealing (SA) is a technique inspired by the formation of crystals in solids during cooling. It is based on the principle that the slower the cooling, the more perfect is the crystal formed. SA has been applied to a vast variety of difficult problems. One classical example is the travelling salesperson problem (Goldberg 1989; Michalewicz 1992).

The algorithm starts with an initial trial solution and its fitness value. The neighbourhood of this solution is searched for better solutions. Once a better solution is found, it becomes the centre of search for the next iteration. The process continues until the acceptable tolerance level is reached. The search distance from the centre point is gradually reduced as the method progresses. The most important characteristic of SA is that it allows traversal in the solution space, which implies that if the algorithm is run for a long enough time, it can hopefully come up with an acceptable (near optimal) solution. The same may not be true for many other iterative algorithms that converge to a local optimum quickly and get stuck there, reducing the chances of improving the existing solution. The direction of possible application of simulated annealing in reservoir systems optimisation is still unclear although the technique has been successfully applied to problems from diverse fields. Kirkpatrick et al. (1983), and Carny (1985) used simulated annealing to solve a classic traveling salesperson problem. Cunha and Sousa (1999) have developed a simulated annealing-based approach to obtain the least-cost design of a looped water distribution network.

2.4.2 Neural Networks

The human brain is an immensely complex network of neurons and it is beyond our capabilities to model the precise structure of brain. However, mathematical models capable of working in a manner similar to the brain have been developed. Such models are known as neural networks (NNs). Sometimes, they are also known as artificial brains.

NNs have evolved from the development of pattern recognition systems for remotely sensed data. A NN consists of many simple processors, each possibly having a small amount of local memory. The processors are interconnected and operate on their local data and on the inputs they receive via the connections. A NN can be trained to learn from certain patterns in input. Once trained, they can be expected to exhibit some capability for generalisation beyond the training data.

Saad et al. (1994) described a NN based disaggregation technique for the operation of multi reservoir systems. First, the reservoir system is aggregated to form a single equivalent reservoir by lumping all the reservoirs together. The NN is then trained using a large number of deterministic optimisations for equally likely sequences of streamflows to give, for an aggregated storage level, the storage level of each reservoir of the system. After the training stage, the optimal policy obtained by SDP for the unique aggregated reservoir can be used as an input to the NN. The output of the NN is in the form of policies for different reservoirs of the system.

Raman and Chandramouli (1996) describe a DP-NN model to obtain operating policies for the Aliyar Dam in India. The input pattern to the NN is the initial storage, inflow, and demand while the output pattern is optimal releases determined by the DP. After the training process, the performance of the trained network is assessed using the simulation model of the system. Applications of fuzzy logic programming in deriving reservoir operating rules have been reported by Shrestha et al. (1996), and by Russell and Campbell (1996)..

2.4.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are general purpose search procedures based on the natural processes and population genetics. EAs include GAs, evolution strategies, evolutionary programming and genetic programming. In 1950s and in the 1960s, several computer scientists began to study evolutionary systems with the idea that the process of natural evolution could be used as a basis of optimisation of complex engineering systems. Rechenberg (1973) introduced a technique based upon evolution strategies to optimise the real-valued parameters for airfoils. Since then, a considerable amount of work has been done on evolutionary computation which is reflected in the numerous applications of EAs to complex engineering systems during the last decade. The popularity of EAs may be attributed to the fact that these algorithms are easy to apply and may be used for problems which are intractable by traditional optimisation algorithms (Michalewicz 1992).

Many variants of EAs exist and there are many hybrid systems which incorporate one or more features of EAs. However, the structure of EAs is very much the same. Each EA begins the search process by starting from a set of potential solutions to the problem. In genetic terminology, these solutions are also known as chromosomes or individuals. The solutions are evaluated by substituting the parameter values into the objective function to give some measure of their fitness. Then a selection process determines the individuals that will enter the new generation. Some members of the new population undergo transformations by means of genetic operators to produce new solutions. The process is repeated for a number of generations and at the end of the algorithm, a reasonably good solution should be obtained. It must be noted that the aim of EAs is not always to find the optimum solution but to find a better solution than can be found by other known techniques in a reasonable computation time. The use of evolutionary computation has been found to be particularly useful for problems with vast search space. Vast search space means the one where it is difficult or nearly impossible to enumerate all the possible solutions to the problem.

EAs are increasingly being used with success to diverse engineering fields like architecture, structural engineering, factory job scheduling, pipe networks, electronic circuit designs, signal processing, robotic controls etc. The application of evolution strategies, evolutionary programming or genetic programming to water resources problems has not been reported so far. The application of GAs has, however, seen some application. A considerable potential exists for application of EAs to water resource problems, and should be utilised.

2.5 Conclusions

In this chapter the basic concepts of mathematical programming techniques were described and application of these techniques to reservoir operation problems reviewed. It is evident from the literature review that a large number of techniques have been developed and applied to the studies of multi reservoir operations problems. A general methodology that can handle reservoir operations without simplifying assumptions is still lacking. There is a need for research into algorithms that have the potential of overcoming the limitations of DP and other DP based algorithms. The DDDP technique described in this chapter will be used in the subsequent chapters to compare the results obtained by using the GA approach. The GA technique is described in the next chapter along with the review of the applications of GA to civil engineering problems in general and water resources problem in particular.

3. GENETIC ALGORITHMS

3.1 Introduction

Despite intensive research carried out during the last three decades, a generic technique for the optimisation of multi reservoir systems is yet to be identified. In recent years, genetic algorithms (GAs) have gained growing popularity among researchers as a robust and general optimisation technique. The approach has been successfully applied to a wide variety of problems from diverse fields. The results of employment of GAs to various difficult optimisation problems have indicated considerable potential in the optimisation of multi reservoir systems. This chapter discusses the basic concepts, theory and working procedures of a GA.

The chapter organisation is as follows. Section 3.2 gives the definition and historical background of the development of GAs. Alternative representation schemes are discussed in section 3.3. The reproduction operators of a GA are described in section 3.4. A coherent review of GA applications in water resources systems in general and reservoir operations in particular is presented in section 3.5. The chapter ends with the concluding remarks.

3.2 Overview of Genetic Algorithms

A genetic algorithm is a technique in which a population of abstract representations of candidate solutions to an optimisation problem are stochastically selected, recombined, mutated, and then either eliminated or retained, based on their relative fitness.

In the *"Origin of Species"*, Darwin (1859) stated the theory of natural evolution. Over many generations, biological organisms evolve according to the principles of natural selection like the "survival of the fittest" to reach some remarkable forms of accomplishment. In nature, individuals in a population have to compete with each other for vital resources such as food and shelter. Because of this, the least adaptable individuals are eliminated from the population while the fittest or the most adaptable individuals reproduce a larger number of offsprings. During reproduction, a recombination of the good characteristics of each parent can produce offsprings whose fitness is greater than either of the parents. After a number of generations, the species evolve spontaneously to become more and more adapted to their environment. Holland (1975) developed this idea in *"Adaptation in Natural and Artificial Systems"* and laid down the first GA. Since then, GAs have developed into a powerful technique for identifying optimal solutions to complex problems. Excellent introductions to GAs are given by Goldberg (1989) and by Michalewicz (1992). Application of GAs to many complex real problems can be found in the works by Davis (1991), Michalewicz (1992) and Dasgupta and Michalewicz (1997).

GAs are a class of artificial intelligence techniques based on the mechanics of natural selection and natural genetics directly derived from the theory of natural evolution. GAs simulate mechanisms of population genetics and natural rules of survival in pursuit of the ideas of adaptation and use a vocabulary borrowed from natural genetics. To surpass the traditional methods, GAs must differ in some very fundamental ways. Goldberg (1989) identifies the following as the significant differences between GAs and more traditional optimisation methods. GAs

- work with a coding of the parameter set, not the parameter themselves;
- search from a population of points, not a single point;
- use objective function information, not derivatives or other auxiliary knowledge;
- use probabilistic transition rules, not deterministic rules.

A GA is a robust method of searching for the optimum solution to a complex problem. It is basically an automated intelligent approach to find a solution to a problem, although it may not necessarily lead to the best possible one. Consider an optimisation problem with 10 parameters with each parameter taking on 100 values. The number of possible combinations of parameters would be 100^{10}. Since the search space is huge, it is not possible to quickly enumerate all the possible solutions. In the past, such problems were tackled by making intelligent guesses about the values of the parameters and a solution obtained by trial and error procedure. But with the advent of GAs, a fairly good solution to such problems can be found within an affordable computing time.

A GA represents a solution using strings of variables that represent the problem. In biological terminology such strings are also known as chromosomes or individuals. Coding components of possible solutions into a chromosome is the first part of a GA formulation.

Each chromosome is a potential solution and comprises of sub-strings or genes representing decision variables which can be used to evaluate the objective function of the problem. The fitness of a chromosome as a candidate solution to a problem is an expression of the value of the objective function represented by it. It is obtained using an evaluation function, which is a link between the GA and the problem to be solved. The fitness is also a function of problem constraints and may be modified through the introduction of penalties when constraints are not satisfied.

A GA starts with a set of chromosomes representing potential solutions to the problem. These chromosomes are combined through genetic operators to produce successively fitter chromosomes. The genetic operators used in the reproductive process are selection, crossover, and mutation. Combination is achieved through the crossover of pieces of genetic material between selected chromosomes. Chromosomes in the population with high fitness values have high probability of being selected for combination with other chromosomes of high fitness. Mutation allows for the random mutations of bits of information in individual genes. The fitness of chromosomes should progressively improve over the generations. The whole GA procedure is allowed to evolve for a sufficient number of generations, and at the end of the evolution process a chromosome representing an optimal (or a near optimal) solution to the problem should be obtained.

3.3 Representation Schemes

The choice of an appropriate coding scheme to represent the potential solutions to a particular problem is the key to success for any GA. To apply GA to a specific problem, an appropriate genetic representation for the solution of the problem must be defined. In early GAs (Holland 1975; DeJong 1975; Goldberg and Kuo 1987) binary coding was used. Real-value coding is now starting to gain recognition and is being used by many researchers. Davis (1991) and Michalewicz (1992) describe many applications of real-value coding.

3.3.1 Binary Representation

Traditionally, binary coding has been used as a representation that can fit all kinds of search spaces. A string of bits can encode integers, real numbers or whatever else is appropriate to the problem. Moreover, binary strings are very simple to operate upon. Each chromosome as a potential solution is represented as a binary string of length

$$L=\sum_{i=1}^{k} m_i \tag{3.1}$$

The first m_1 bits map into a value from the range $[a_1, b_1]$, the next group of m_2 bits maps into a value from the range $[a_2, b_2]$ and so on. The last group of m_k bits map into a value $[a_k, b_k]$, k being the number of decision variables. Then, choosing the parameter values randomly throughout the search space, a set of initial potential solutions, also known as chromosomes, is created. The GA evaluates the initial solutions and computes a measure of fitness of each string in the population. This is achieved by decoding binary strings into parameter values, substituting them into the objective function and computing the value of the objective function for each of the strings.

Binary coding has been found to perform well on a variety of problems but it may not be appropriate for problems where the number of variables is large or where a high precision is required. Bit strings are used with some type of decoding function but that has difficulty ensuring even coverage of the search space because of the way in which the real values are encoded. Additionally, the use of bit strings causes non-uniform effects because mutating the leading bits of strings has a greater impact than mutating the tail bits.

3.3.2 Gray Representation

In Gray coding, any two adjacent points in the problem space differ by one bit only (Hollstein 1971). An increase of one step in the parameter value corresponds to a change of a single bit in the code. Table 3.1 shows the correspondence between coded substrings for binary and Gray coding schemes. It can be observed from the table that in Gray coding only 1 bit changes between adjacent substrings of 001, 011, and 010 etc. On the other hand, adjacent substrings in binary coding may differ by any number of bits. For example, adjacent substrings 011 and 100 differ by 3 bits.

In a binary coded string, a single mutation in the most significant bit may significantly change the number. Also, the effect of mutating tail bits is more than that of mutating leading bits. In a Gray coded string, the bitwise mutation operator causes less disruption to the solution because the effect of flipping a single bit in a string is small most of the time. Gray coding can therefore increase a mutation operator's chance of making incremental improvements and thus enhance the performance of GA.

Table 3.1 Correspondence between binary and Gray coding schemes

Binary Code	Gray Code	Decimal Value
000	000	0
001	001	1
010	011	2
011	010	3
100	110	4
101	111	5
110	101	6
111	100	7

3.3.3 Real-value Representation

A problem specific representation, can be used to give more coherence and efficiency to the algorithm. For the problems where the function arguments are integer or real values, integer or real-value representation may be used. The integer or the real-value representation has the property that two points close to each other in the representation space are also close in the problem space and vice versa.

In real-value representation, the genes of chromosomes are initially allocated values randomly within the feasible limits of the decision variables. Hence, no time is wasted in decoding the binary values into decimal values and in mapping them within the desired range. A significant advantage of this type of coding scheme is that no discretization of the decision variable space is required. In addition, the real-value coding is capable of representing quite large domains. On the other hand, binary coding sacrifices precision with an increase in domain size for a fixed string length. The precision of binary coding can be extended by introducing more bits in the string, but this slows down the algorithm considerably. Michalewicz (1992) indicates that for real-valued numerical optimisation problems, real-value representation outperforms binary representation because it is more consistent, more precise, and leads to faster execution.

3.4 Reproduction Operators

The reproduction operation is the basic engine of Darwinian natural selection and survival of the fittest. In a GA, the reproduction mechanism comprises of the following three operators: selection, crossover, and mutation.

3.4.1 Selection

Selection is the procedure by which the chromosomes from one generation are chosen for the next generation. The process is carried out using an appropriate selection scheme. Alternative selection schemes are described below.

3.4.1.1 Proportional Selection Scheme

The most popular and the commonly used selection method has been fitness proportional selection (Goldberg 1989). The selection of chromosomes in a simple GA is based directly on its fitness which is computed using the objective function of the problem. Given a population of chromosomes, the probability of a particular chromosome passing its gene to the next generation is directly proportional to its fitness. The probability p_i of selection of chromosome i, to go into the next generation is given by

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \tag{3.2}$$

where f_i = fitness of string i and N is the population size. Such a scheme is also known as roulette wheel selection. A roulette wheel for a population size of 4 is shown in Figure 3.1. The slots in the roulette wheel are sized according to the fitness of chromosomes 1-4. The roulette wheel is spun 4 times, and each time a single chromosome is selected for the new population. In the example considered, the probability of selection of the best chromosome (chromosome 4) is 55% whereas the worst chromosome (chromosome 1) has a selection probability of only 5% (Figure 3.1). At the end of four spins, the chances of the best chromosome having more than a single copy are quite high. On the other hand, the worst chromosome is unlikely to enter the next generation.

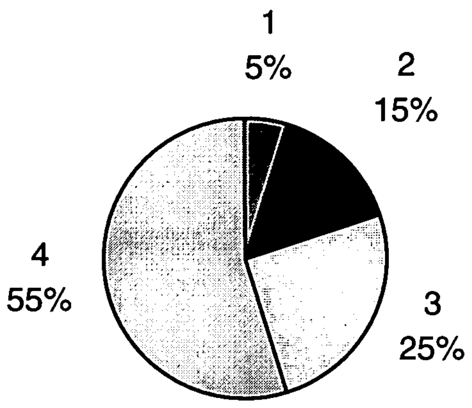


Figure 3.1 Roulette wheel selection

The fitness proportional selection scheme is likely to lead to premature convergence due to lack of selective pressure. Whitley (1989) identifies that population diversity and selective pressure are two important issues in the genetic search process. Population diversity means that the already discovered good individuals are exploited while still exploring the search space for promising new areas. The diversity in the population depends upon the selective pressure which determines the degree to which the best individuals are favoured. These factors are strongly related; a high selective pressure may lead to a rapid convergence, often to a sub-optimal point. On the other hand, low selective pressure may unnecessarily delay the process of reaching the desired point in the search space. Therefore an ideal selection scheme would be one which maintains the selective pressure and the population diversity at the same time. Yang and Soh (1997) provide an interesting discussion on this topic.

3.4.1.2 Rank Selection Schemes

In rank selection schemes, the chromosomes are selected according to their rank rather than actual fitness values. Such approaches have been shown to improve GA performance but have apparent drawbacks. They ignore the information about the relative fitness of different chromosomes and treat all cases uniformly regardless of the magnitude of the problem. On the other hand, these schemes control the selective pressure better and provide the search with a greater focus.

The rank selection schemes operate by sorting the population according to the fitness and then assigning a probability of selection based upon the rank. A constant selection differential is thus maintained between the worst and the best individuals. In linear ranking schemes, the probability of selection of a chromosome in a single selection is assigned according to the following linear assignment function.

$$p(n) = q - (n-1) \times r \qquad (3.3)$$

where $p(n)$ is the probability of an individual ranked in position n to be selected in a single selection, and q and r are user defined parameters which control the selective pressure. Finally, proportional selection is performed based on the probabilities of selection assigned according to (3.3).

3.4.1.3 Tournament Selection Scheme

The tournament selection is a very aggressive type of selection scheme. Goldberg and Deb (1991) observed that it eliminates random noise from the selection process and improves the efficiency of the GA search algorithm. Two or more chromosomes are randomly picked from the population and then the best one is selected from this group for further genetic processing. This procedure is repeated appropriate number of times to obtain the required number of chromosomes for the next generation. The approach had originally been developed with groups of two individuals and was called binary tournament selection, but larger groups lead to greater diversity and a smoother progression to a solution.

3.4.2 Crossover

Crossover is a distinguishing feature of a GA. Through crossover, genetic information is exchanged between parents to produce offsprings. The offsprings then replace parents in the new population in order to keep the population size constant. If the mating does not take place, the parent strings survive into the next generation. The general theory behind the crossover operation is that by exchanging important genes between the parents that perform well, the GA attempts to produce children that have the best characteristics from both parents and therefore perform even better than the parents.

The crossover operator creates variation in the population by producing offsprings that consist of genes taken from each parent. Crossover occurs with some specified probability, usually in the range of 0.5 - 1.0, which controls the number of parents that undergo crossover. A crossover probability of 0.7 implies that crossing over pairs of chromosomes from the previous population creates about 70% of the chromosomes in the new population.

The other 30% or so pass to the next generation without being crossed over. Goldberg (1989) describes several methods of performing the crossover operation.

3.4.2.1 One Point Crossover

The most straightforward method of crossover is one point crossover. Pairs of chromosomes are randomly selected and a crossover point c is also selected along the length L of the chromosomes. Two new chromosomes are created by swapping all genes between position c and L inclusively (Figure 3.2). An important feature of one point crossover is that it can produce children that are radically different from their parents. With one point crossover, the head and tail of one chromosome cannot be passed together to the offspring. If both the head and tail contain good genes, the offsprings may not share those genes at the same time. This drawback can be overcome by using a two point crossover.

3.4.2.2 Two Point Crossover

Two chromosomes are randomly selected as in one point crossover. This operator is similar to one point crossover except that two cut points rather than one are selected at random along the length of the chromosome (Figure 3.2). The current genes between the two cut points are swapped thus creating two children. The parents are then replaced in the new population by the children. This type of crossover leads to greater diversity than the one point crossover.

3.4.2.3 Uniform Crossover

In uniform crossover, individual genes are exchanged rather than a block of genes. Two parents are randomly selected to take part in crossover. Each gene in the offspring is created by copying the corresponding gene from one or the other parent. For each gene or bit in the first offspring, the operator performs a probability check and decides which parent will contribute its gene in that position. The second offspring receives the gene from the other parent (Figure 3.2).

The use of uniform crossover is likely to lead to greater diversity in the population than either the one point or the two point crossover. Syswerda (1989) carried out experiments to evaluate the performance of uniform crossover operator. It was observed that the ability of the operator to combine the genes irrespective of their location in the chromosome outweighs the disturbance that it may cause when used on radically dissimilar chromosomes.

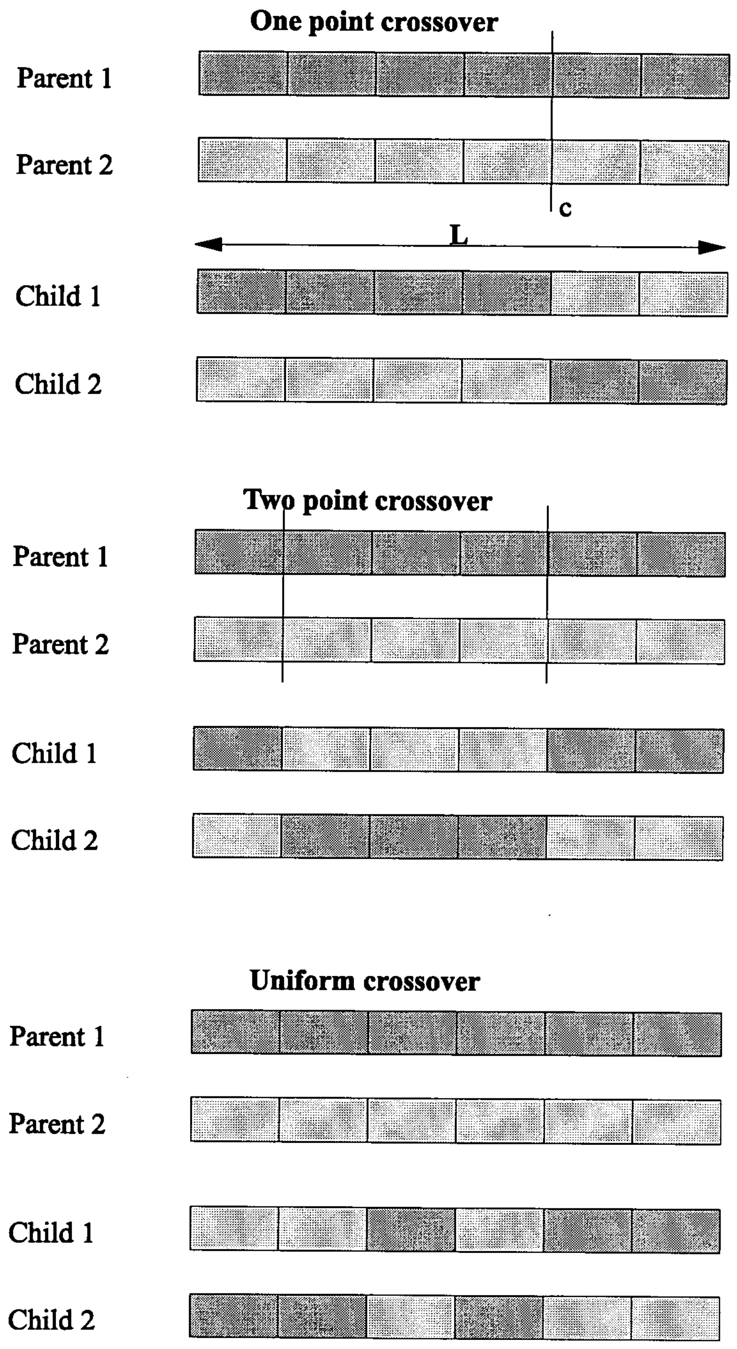


Figure 3.2 Approaches to crossover

3.4.3 Mutation

Mutation plays a vital role in genetic optimisation introducing new genetic material in a heterogeneous population. The mutation operator can be useful in reintroducing diversity in a population that may be tending to converge prematurely. Mutation occurs with some specified probability which is usually very low (0.1 - 0.001) for each bit in the strings. In binary coding, the bitwise mutation operator changes the value of the bit to the opposite value i.e. 0 to 1 or 1 to 0. Figure 3.3 illustrates this scheme. Various mutation operators are in use these days. The most important types are uniform mutation and non uniform mutation.

3.4.3.1 Uniform Mutation

The most common type of mutation used in integer or real-value representation is the uniform mutation. This operator requires an individual *y* and produces a child *y*\`. The operator selects a random component $j \in (1, n)$ of the vector $y = (y_1,...,y_j,...y_n)$ and produces $y` = (y_1,...,y`_j,...y_n)$ where y_j is a random value of the component *j* between the upper and lower bounds.

The value of component $y`_j$ is obtained as follows.

$$y'_j = \begin{Bmatrix} y_j + \beta & if \quad r = 0 \\ y_j - \beta & if \quad r = 1 \end{Bmatrix} \tag{3.4}$$

where *r* is a randomly generated binary digit, and β is small change in the value of the component *j*.

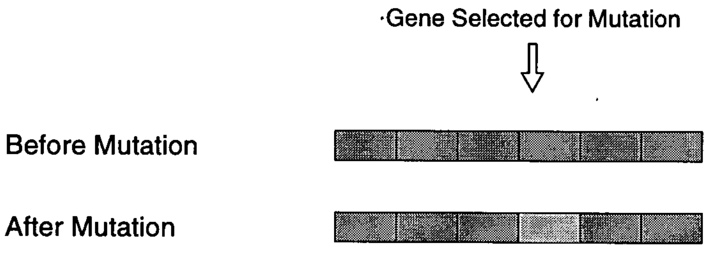


Figure 3.3 Mutation process

3.4.3.2 Non Uniform Mutation

Non uniform mutation operator selects a gene to mutate with a specified probability, and then replaces it with a value between predefined upper and lower limits. The values of upper and lower limits vary as the GA run progresses. This operator has been found useful in searching the decision space fairly uniformly during early stages of run and locally during the later parts of the run (Michalewicz 1992). The operator has the effect of further improving already found good solutions during the later part of the run without causing much disruption.

The application of genetic operators to the current population of chromosomes produces a new population. The whole process continues for a number of generations and an optimal or a near optimal solution should be obtained in a reasonable computation time.

3.5 Application of GAs to Engineering Problems

This section presents a brief review of GA applications to engineering problems with particular emphasis on water resources problems. In recent years researchers have used GAs for solving complicated engineering problems. Goldberg and Kuo (1987) applied a GA to the optimisation of the operation of a steady state serial gas pipeline. Their pioneering work paved the way for application of GAs to more complicated engineering problems.

Wang (1991) applied a GA to the calibration of a conceptual rainfall-runoff model for a particular catchment. Conceptual rainfall-runoff models usually consist of a number of parameters. Their model had seven calibration parameters, the values of which were optimised by minimising the sum of squares of differences between computed and observed discharge. Of the 10 runs of the GA model, each starting from a different set of randomly selected initial points in the search space and with 5000 objective function evaluations, eight runs were able to locate the global peak. The value obtained from other two runs was only marginally inferior to the global optimum. Similar work has been reported by Franchini (1996), who used a local search method called Sequential QP (SQP) in combination with GA for the calibration of a conceptual rainfall-runoff model. It was concluded that GA-SQP algorithm is a practical and efficient technique and performs better than a simple GA.

There have been several applications of GAs to pipe network problems. Goldberg and Kuo (1987) were the first to use GAs for pipeline optimisation. Murphy et al. (1993) developed a methodology for optimisation of a water supply network using a simple GA. The objective was to find the combination of pipe sizes that minimised the cost of water distribution network. Simpson et al. (1994) compared the performance of complete enumeration, non-linear programming (NLP) and a GA for an example pipe network. They concluded that the GA was capable of finding acceptable solutions within affordable computing time. For the example considered, the GA was not as fast as NLP but it was faster than the direct enumeration.

Davidson and Goulter (1995) used GAs to optimise the layout of a branched rectilinear network, such as a natural gas or water distribution system. They demonstrated that GA is able to generate better solutions than a heuristic technique. An improved GA has been developed by Dandy et al. (1996) for pipe network cost optimisation problem and was found to perform better than the traditional optimisation methods and a simple GA. The solution found by improved GA is the lowest cost design for the New York City water supply network yet presented in the literature for that particular problem. Savic and Walters (1997) describe the development of the computer model GANET for the least cost design of a water distribution networks, again demonstrating that in certain cases GAs may yield better results than other optimisation techniques.

Halhal et al. (1997) have described a multiobjective optimisation approach, using capital cost and benefit as dual objectives to, the problem of network rehabilitation. They used a structured messy GA (SMGA) which has some additional features like variable string length which increases during the evolution of designs. They compared the performance of SMGA with a standard GA, concluding that SMGA was much better for a large network.

Ritzel et al. (1994) solved a multiple objective groundwater pollution problem using GAs. Cieniawski et al. (1995) used GAs to solve a multi objective groundwater monitoring problem. The work by Cieniawski et al. (1995) dealt with the optimal location of a network of groundwater monitoring wells under conditions of uncertainty. The GA approach allowed the maximisation of reliability of well location and minimisation of contaminated area, separately yet simultaneously. McKinney and Lin (1994) have also used GAs for the solution of groundwater management models.

There have been many applications of GAs to other civil engineering problems. An increasing number of researchers are using GAs to solve civil engineering problems which are difficult with traditional techniques. Koumousis and Georgiou (1994) applied GA to the optimisation of steel truss roofs. Soh and Yang (1996) used GAs in combination with fuzzy logic for structural shape optimisation problem. Rajeev and Krishnamoorthy (1992; 1997), Jenkins (1991; 1992), and Adeli and Cheng (1993; 1994) have also reported applications to structural engineering design problems. A methodology based on GAs has been developed by Li and Love (1998) for optimising the layout of construction site level facilities such as warehouses, offices, various workshops and batch plants. Navon and Mcrea (1997) used the GA approach for selection of construction robots. Feng et al. (1997) applied GA to the problem of cost-time trade-offs in construction projects. Applications of GA to transportation engineering problems have been described by Ng (1995), Fwa et al. (1996), Liu et al. (1997), and Cheu et al. (1998).

GAs have so far had very little application in reservoir systems optimisation. Fahmy et al. (1994) used GAs for optimisation of reservoir systems operation. They compared the performance of a GA with that of DP for a hypothetical system and concluded that GAs had potential in application to large river basin systems. Esat and Hall (1994) solved the four reservoir problem using GA. The paper by Esat and Hall (1994) showed the significant potential of the GAs in water resources systems optimisation, and clearly demonstrated the advantages of GAs. Oliveira and Loucks (1997) used a GA for the determination of effective operating policies for multi reservoir systems. Significant benefits were perceived to lie in the freedom afforded by GAs in the definition of operating policies and their evaluation.

3.6 Conclusions

In this chapter the essentials of the GA approach have been described. An introduction to different representation schemes, and genetic operators was provided. A short review of the application of GAs to water resources problems was also presented. It is concluded that a wide variety of problems from diverse fields have been solved using the GA technique, and an increasing number of researchers are using the technique to solve difficult engineering problems. There have been, however, only a limited number of applications of GAs to reservoir operation problems. The GA approach introduced in this chapter is applied in the subsequent chapters to optimise the operation of multi reservoir systems.

4. APPLICATION OF GENETIC ALGORITHMS TO THE FOUR RESERVOIR PROBLEM

4.1 Introduction

The primary objective of this chapter is to explore the potential of alternative GA formulations in application to reservoir systems. The theory, and a brief review of GA applications to engineering problems, have been presented in the previous chapter. The review of literature indicates that GAs have had limited application in water resources planning and management. Applications have been reported in the literature for water distribution network problems, groundwater pollution problems and calibration of rainfall-runoff models, but there have been only a few applications to reservoir systems. GAs may be set up in a number of ways, but as yet there is little guidance in the literature on the type of formulation most appropriate for reservoir systems. This gap is addressed through consideration of the application of GAs to the well-known four reservoir problem, which has become a benchmark for water resources system optimisation algorithms. The results produced by GA model have been verified by comparing to those obtained from the discrete differential dynamic programming (DDDP) and linear programming (LP) models.

The rest of this chapter is organised as follows. Section 4.2 gives the description of the four reservoir problem. Section 4.3 describes the procedure for formulation of a GA to solve a multi reservoir optimisation problem. Different representation schemes are described in section 4.4. In section 4.5, the LP solution to the four reservoir problem is presented. The development of DDDP computer code for the four reservoir problem is described in section 4.6, and in section 4.7, the procedure for obtaining a solution to a problem using a GA is outlined. Evaluation of alternative GA formulations has been carried out in section 4.8. In section 4.9, the four reservoir problem with an extended time horizon is solved. The modified four reservoir problem is solved in section 4.10, and the application of the GA to a ten reservoir problem is demonstrated in section 4.11. Conclusions are presented in section 4.12.

4.2 The Four Reservoir Problem

This section gives the details of the four reservoir problem which was formulated by Larson (1968). The problem was first solved by Larson (1968) using LP and by state incremental dynamic programming (IDP). Heidari et al. (1971) solved the same problem using the DDDP approach. Chow et al. (1975) used the four reservoir problem as one of the test cases for estimation of execution time and memory requirements of DDDP. Murray and Yakowitz (1979) illustrated the differential dynamic programming (DDP) algorithm using the four reservoir problem and Nopmongcol and Askew (1976) tested multi level IDP on the same problem. The four reservoir problem has been chosen for solution by GA so that the performance of the GA could be evaluated against a known global optimum, and that sensitivity analyses could be performed. The same problem has been previously tackled using a GA by Esat and Hall (1994). Little detail was given by Esat and Hall (1994) on their GA formulation. The approach outlined in this study is different from the one used by them.

The system consists of four reservoirs having series and parallel connections, as shown in Figure 4.1.

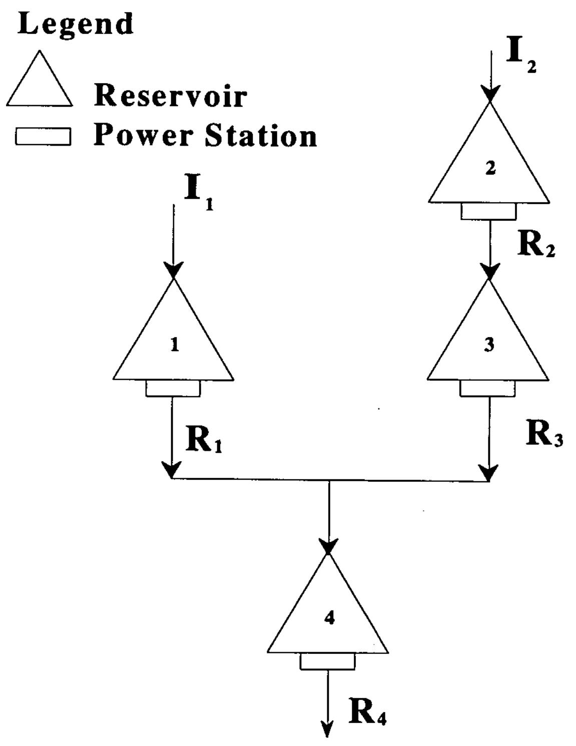


Figure 4.1 The four reservoir system

Reservoir releases from the system are used for hydropower generation and irrigation. Hydropower generation is possible from each reservoir, and releases from reservoir 4 can also be diverted for irrigation after passing through the turbines. The objective is to maximise hydropower and irrigation benefits from the system over 12 two-hour operating periods. The inflows I_1 and I_2 to the reservoirs 1 and 2 during all time steps are 2 and 3 units respectively. The state variables for the problem are the storages S_i contained in the reservoirs i =1, 4 expressed in standardised units. The decision variables are the releases $\mathbf{R}_i\mathbf{(t)}$, i = 1, 4; and t = 1, 12.

The maximum water level in each reservoir is limited by flood control considerations, thereby leading to the following constraints on the storage during any operating periods.

$$0 \le S_1, S_2, S_3 \le 10 \qquad (4.1)$$

$$0 \le S_4 \le 15 \qquad (4.2)$$

The initial contents of each of the four reservoirs is 5 units whereas the target ending storages are 5 units for reservoir 1-3 and 7 units for reservoir 4. The turbine capacities determine the maximum flows from each reservoir and conservation requirements downstream set the minimum release requirements as described below.

$$0 \le R_1 \le 3 \qquad (4.3)$$

$$0 \le R_2, R_3 \le 4 \qquad (4.4)$$

$$0 \le R_4 \le 7 \qquad (4.5)$$

These constraints apply in all time steps.

The dynamic behaviour of the system at any stage t = 1, 12 is described by the following mass balance equation.

$$S_i(t+1) = S_i(t) + I_i(t) + MR_i(t) \qquad (4.6)$$

where $\mathbf{S}_i$(t) = vector of reservoir storages at time t in reservoirs $i=1,\ n$; $\mathbf{I}_i$(t) = vector of reservoir inflows in time period t to reservoirs $i=1,\ n$; $\mathbf{R}_i$(t) = vector of reservoir releases in time period t from reservoirs $i = 1, n$; and $\mathbf{M} = n \times n$ matrix of indices of reservoir connections.

The matrix M has -1 along the diagonal and +1 in the position ith column and jth row if the release from reservoir i goes into reservoir j. The rest of the martrix elements are zero. From this consideration, the matrix M for the four reservoir system can be written as

$$M = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & -1 \end{bmatrix} \tag{4.7}$$

The performance criterion also includes a penalty if the storage falls below the target level d_i, $i = 1, 4$ at the end of 12 operating periods. The following penalty function used by Larson (1968) has been adopted in this study.

$$g_i[s_i(13),\, d_i] = \begin{Bmatrix} -40[s_i(13) - d_i]^2 & if\ s_i(13) \le d_i \\ 0 & if\ s_i(13) > d_i \end{Bmatrix} \tag{4.8}$$

The penalty function is not needed in LP since the boundary conditions can always be set up as constraints in the LP model.

The performance criterion to be maximised is the sum of returns due to power generated by the four reservoirs and the return from the diversion of $R_4(k)$ to the irrigation project. The objective function may be expressed as:

$$Maximise\ \ F = \sum_{i=1}^{4}\sum_{t=1}^{12} b_i(t).\, R_i(t) + \sum_{t=1}^{12} b_5(t).\, R_4(t) + \sum_{i=1}^{4} g_i[s_i(13),\, d_i] \tag{4.9}$$

where F is the total benefit obtained from the system after 12 operating periods and $b_i(t)$ is the unit return due to activity $i = 1, 5$ for the operating period t. The coefficients $b_i(t)$ are based on the values provided by Larson (1968) and Heidari et al. (1971).

4.3 Formulation of GAs

To solve the four reservoir problem using a GA, it is necessary to construct a chromosome representing all four reservoirs in all twelve time steps. Since the objective function is based on reservoir releases in each time step, releases should be the decision variable on which the GA is based. With four reservoirs and twelve time steps, there are thus 48 discrete variables to be represented in a GA. Each of these is considered as a gene and a chromosome comprising of these genes represents a possible solution to the problem.

Each chromosome consists of genes, which are made up of alleles. In binary and Gray coding, an allele represents a binary bit (0 or 1), whereas in real-value coding it represents a real or integer value of the variable defined between the upper and lower bounds. For the four reservoir problem, reservoir releases are to be considered as integer quantities. This is a definition of the problem and not a limitation of GAs. GAs work equally well with non integer decision variables. In binary and Gray representation, 3 alleles or digits are required to cover the range of releases for the problem. With this approach the total length of chromosome in binary coding is 144. Esat and Hall (1994) used only binary coding and refer to a chromosome length of 16. It is unclear how these chromosomes were made up. In real-value coding a gene consists of only a single allele, and the length of a chromosome in terms of number of genes is 48 (12 time steps and four reservoirs).

The manner in which genes are grouped within the chromosome is also important. There are two basic approaches. One would be to group releases by time step, such that the chromosomes contained 12 groups of 4 genes representing the releases from each reservoir in a particular time step. The alternative is to have 4 groups of 12 genes, with each group containing the time series of releases from an individual reservoir. The former approach is used in this study as it keeps more closely related material together.

4.4 Representation Schemes

This section describes alternative representation schemes. GAs can be set up in a number of ways as outlined in the previous chapter but there is no standard procedure of doing so. Therefore, alternative coding schemes: binary, Gray and real-valued have been used for this problem.

4.4.1 Binary Representation

In binary representation, the decision variables of the problem are encoded as sub-strings of binary bits. These sub-strings of decision variables are concatenated to form longer strings or chromosomes representing a solution. Since the objective function for the four reservoir problem is based on reservoir releases in each time step, releases should be the decision variables on which the GA is based.

Each decision variable for this problem is encoded as a binary number consisting of three digits as shown in Table 4.1. The use of a 3 bit binary coding to represent release for each stage allows the decision variable to be discretized into $2^3 = 8$ points. This is sufficient for the range of releases defined for the four reservoir problem. With this approach, the total length of sub-string representing a potential solution becomes $12 \times 4 \times 3 = 144$.

Table 4.1 Coding of releases for the four reservoir problem

Binary Code	Gray Code	Decimal value	Release 1	Release 2	Release 3	Release 4
000	000	0	0.00	0.00	0.00	0.00
001	001	1	0.43	0.57	0.57	1.00
010	011	2	0.86	1.14	1.14	2.00
011	010	3	1.29	1.71	1.71	3.00
100	110	4	1.71	2.29	2.29	4.00
101	111	5	2.14	2.86	2.86	5.00
110	101	6	2.57	3.43	3.43	6.00
111	100	7	3.00	4.00	4.00	7.00

For the four reservoir problem, the chromosomes must consist of sub-strings representing releases from each reservoir in each time step. This formulation was not used by Esat and Hall (1994). They refer to strings of length 16, which could represent releases for a single stage only. The strings used by them cannot represent a solution to the problem, although this is the basic requirement for obtaining a solution to a problem by GA. It appears that they have not included every decision variable of the problem in their formulation. It remains unclear how these strings of length 16 were made up.

4.4.2 Gray Representation

Standard binary coding permits large jumps in the value of variables, as the effect of mutating the tail bits is large compared to mutating the leading bits. This problem can be overcome to some extent by the use of Gray coding (Goldberg 1989) in which the binary representation of a variable value changes by only one binary digit between generations. Discretization of the decision variable space is required with binary and Gray coding. The length of sub-string required to represent a variable depends upon the level of precision required. String length can be increased to achieve desired accuracy in the solution and satisfactory mapping of the variable space can normally be achieved.

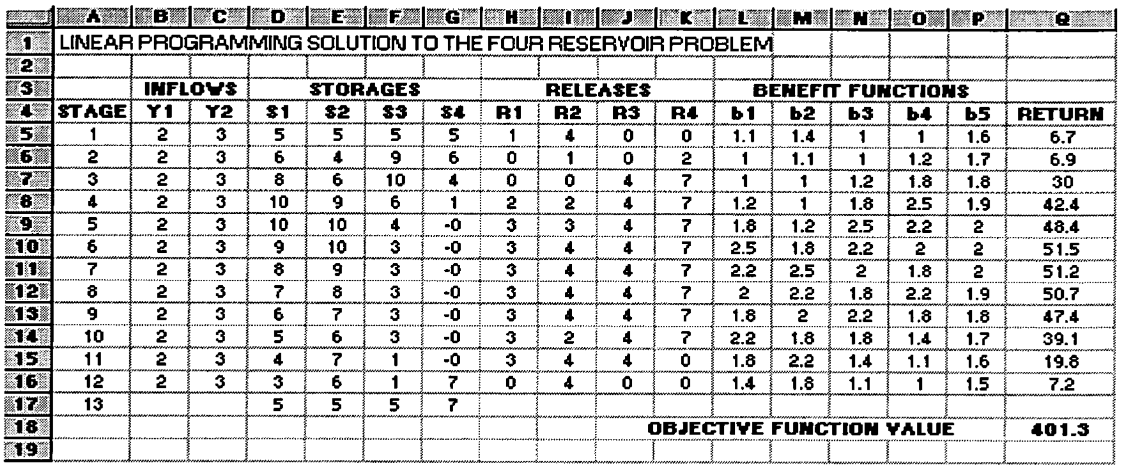
Gray code representation can increase a mutation operator's chance of making incremental improvements and thus enhance the performance of GA. The bitwise mutation operator causes less disruption to the solution in Gray coding because the effect of flipping a single bit in a string is small most of the time. However, occasionally Gray coding can lead to a big change and this is the drawback associated with it. For example, a single mutation of the left most bit in 000 changes the decimal value to 7 (100) and vice versa.

4.4.3 Real-value Representation

An alternative approach to the formulation of GA is to use a representation appropriate to the components of the problem. Real-value chromosomes have been used with success by various authors (e.g., Michalewicz 1992; Davis 1991; Oliviera and Loucks 1997). In real-value representation, the genes consist of single allele only and are themselves the parameter values. The chromosome representing the solution to the four reservoir problem thus consists of 48 genes with each gene initially allocated a random value between the upper and lower bounds of the variables of the problem. With real-value coding, discretization of decision variables is not required.

4.5 Solution Procedure Using LP

Since the objective function and the constraints for the four reservoir problem are linear, it is solvable by LP. The solution to the problem was obtained in a spreadsheet using Microsoft Excel solver (MES). The spreadsheet set up to solve the four reservoir problem using LP is shown in Figure 4.2. Column A of the spreadsheet shows the time steps of the problem. The inflows to reservoir 1 and 2 are shown in columns B and C respectively. The storges in reservoirs are obtained using the continuity equation, and are shown in columns D to G. Column H to K contains the decision variables (releases) of the problem. The benefit functions are shown in columns L to P. Using these benefit functions and the releases made from the reservoirs, the return at each time step is computed. This is shown in column Q of the spreadsheet.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	LINEAR PROGRAMMING SOLUTION TO THE FOUR RESERVOIR PROBLEM																
2																	
3		INFLOWS		STORAGES				RELEASES				BENEFIT FUNCTIONS					
4	STAGE	Y1	Y2	S1	S2	S3	S4	R1	R2	R3	R4	b1	b2	b3	b4	b5	RETURN
5	1	2	3	5	5	5	5	1	4	0	0	1.1	1.4	1	1	1.6	6.7
6	2	2	3	6	4	9	6	0	1	0	2	1	1.1	1	1.2	1.7	6.9
7	3	2	3	8	6	10	4	0	0	4	7	1	1	1.2	1.8	1.8	30
8	4	2	3	10	9	6	1	2	2	4	7	1.2	1	1.8	2.5	1.9	42.4
9	5	2	3	10	10	4	-0	3	3	4	7	1.8	1.2	2.5	2.2	2	48.4
10	6	2	3	9	10	3	-0	3	4	4	7	2.5	1.8	2.2	2	2	51.5
11	7	2	3	8	9	3	-0	3	4	4	7	2.2	2.5	2	1.8	2	51.2
12	8	2	3	7	8	3	-0	3	4	4	7	2	2.2	1.8	2.2	1.9	50.7
13	9	2	3	6	7	3	-0	3	4	4	7	1.8	2	2.2	1.8	1.8	47.4
14	10	2	3	5	6	3	-0	3	2	4	7	2.2	1.8	1.8	1.4	1.7	39.1
15	11	2	3	4	7	1	-0	3	4	4	0	1.8	2.2	1.4	1.1	1.6	19.8
16	12	2	3	3	6	1	7	0	4	0	0	1.4	1.8	1.1	1	1.5	7.2
17	13			5	5	5	7										
18													OBJECTIVE FUNCTION VALUE				401.3
19																	

Figure 4.2 Spreadsheet to solve the four reservoir problem

The MES requires that the target cell that is to be maximised or minimised is specified. For the four reservoir problem the target cell contains the value of the total return obtained from the operation of the system for 12 time steps. The target cell is shown in the bottom right corner of the spreadsheet (cell Q18). The constraints of the problem can also be easily set up in MES. The cells containing the decision variables of the problem are specified, and these cells (column H to column K) are adjusted by the MES until an optimal solution is found. After solving the problem, a report that summarises the results of a successful solution process can be created. The report produced by the MES for the four reservoir problem is shown below.

Microsoft Excel 5.0 Answer Report
Worksheet: [4R_12.XLS]Sheet1
Report Created: 15/8/99 12:27

Target Cell (Max)

Cell	Name	Original Value	Final Value
$Q$18	RETURN	203	401.3

Adjustable Cells

Cell	Name	Original Value	Final Value
$H$5	R1	2	1
$I$5	R2	2	4
$J$5	R3	2	0
$K$5	R4	2	0
$H$6	R1	2	0
$I$6	R2	2	1
$J$6	R3	2	0
$K$6	R4	2	2
$H$7	R1	2	0
$I$7	R2	2	0
$J$7	R3	2	4
$K$7	R4	2	7
$H$8	R1	2	2
$I$8	R2	2	2
$J$8	R3	2	4
$K$8	R4	2	7
$H$9	R1	2	3
$I$9	R2	2	3
$J$9	R3	2	4
$K$9	R4	2	7
$H$10	R1	2	3
$I$10	R2	2	4
$J$10	R3	2	4
$K$10	R4	2	7

$H$11	R1	2	3
$I$11	R2	2	4
$J$11	R3	2	4
$K$11	R4	2	7
$H$12	R1	2	3
$I$12	R2	2	4
$J$12	R3	2	4
$K$12	R4	2	7
$H$13	R1	2	3
$I$13	R2	2	4
$J$13	R3	2	4
$K$13	R4	2	7
$H$14	R1	2	3
$I$14	R2	2	2
$J$14	R3	2	4
$K$14	R4	2	7
$H$15	R1	2	3
$I$15	R2	2	4
$J$15	R3	2	4
$K$15	R4	2	0
$H$16	R1	2	0
$I$16	R2	2	4
$J$16	R3	2	0
$K$16	R4	2	0

4.6 Solution Procedure Using DDDP

The memory and execution time needed for DP are huge for problems of moderate size and complexity. Consider a two reservoir system with storage discretized into 100 states each. The number of possible combinations of states in each reservoir would be 100 x 100 per stage. For a two reservoir system, the combinations become 100^4 = 100 million requiring huge computational resources. For this reason, it could be safely concluded that DP becomes computationally bounded for problems of moderate dimension and complexity. One of the highly regarded methods of alleviating the dimensionality problem associated with DP is DDDP. In DDDP, the recursive equation of DP is solved for only a limited number of states. Heidari et al. (1971), Chow and Cortes-Rivera (1974), and Turgeon (1982) tested three states at each stage of the problem. Higher number of states may be tested at each stage but this leads to increased computational complexity. With only three states to be considered in the current time step and three in the next time step, there are consequently only nine possible passages from one stage to another. The amount of memory required to store the solutions is also drastically reduced compared to DP. DDDP therefore leads to a computational procedure that has significant advantages in terms of memory and computational requirements. These savings are achieved without any compromise in the generality of problems that can be treated.

In this research, a computer code based on the DDDP approach has been developed for the four reservoir system so that the comparisons of solutions with LP and GA could be made. The DDDP computer code used to solve the four reservoir problem is written in Microsoft Visual C++, and is presented in appendix C of this thesis. For the four reservoir problem, there are 21,296 combinations of reservoir storage states at each stage. If the release increment is chosen as one unit, then the total number of combinations of decisions that must be tested with DP at each state of each stage is 4 x 5 x 5 x 8 = 800. Due to huge computing requirements, the application of DP to this problem is ruled out. The problem is instead solved using DDDP.

The application of DDDP to the four reservoir problem requires that the problem be divided into states and stages. The state corresponds to the storage in the reservoir and the stage represents the time period. Further, each stage must have a finite number of states associated with it. For the development of reservoir states, the storage volume of each reservoir has been discretized in increments of 1 unit each. This gives 11 states for reservoirs 1-3, and 16 states for reservoir 4. The release increment is chosen as 1 unit since the releases are allowed to assume integer values only. To initiate the DDDP procedure, trial state trajectories are required for each reservoir. The trial state trajectories were obtained by setting releases to inflows in all time steps. The exeception to this occurs in the last time step when the boundary conditions have to be taken into account. After the determination of trial trajectory, a corridor is constructed around it which specifies the limiting values of the state variables within which the optimization is to be carried out. For this problem, one state below and one above the trial state trajectory were used to define the corrodor. This results in a significant reduction in the execution time as only a small subset of all feasible states at each stage is analysed. An improved trajectory that optimises the objective function given by (4.9) is obtained after each iteration and used as a trial trajectory for the next iteration.

The procedure continues until no further improvement in the value of objective function is obtained. With DDDP, a solution identical to that obtained from LP was produced. The state trajectories produced by DDDP and by LP for the four reservoir problem are presented in section 4.8. In the same section, the GA solution is also presented.

Clearly, there are advantages associated with the DDDP approach in terms of computing resources but as the dimensionality of the system increases the method too becomes computationally bounded. There are a few drawbacks of the approach even for small systems. Stringent conditions must be satisfied to achieve optimal solutions (Turgeon 1982). The choice of state increments is also important. Using a large state increment could result in local optimal solutions. On the other hand, if the state increment is too small, a large number of unnecessary computations are performed. Furthermore, an initial trial state trajectory is needed for each reservoir to initiate the DDDP procedure, the determination of which could be cumbersome. Also, if the initial trial trajectory is far from the optimal region, then the convergence may be to a local solution. With GA, however, no trial state trajectories are required. This is another indicator of the robustness of the search procedure that underlies the GA approach. Further, the development of a general purpose DDDP code is complicated and requires significant effort. On the other hand, a GA is easy to program in generic form. Many general purpose commercial GA packages are available and can be easily applied to multi reservoir optimization problems. This could be regarded as an encouraging sign for the practitioners in the field, who are not always provided with well documented and decipherable techniques (Yeh 1985).

4.7 Solution Procedure Using GA

4.7.1 The Solution Procedure

The steps involved in the genetic optimisation of the problem under consideration are as follows.

1. Initialisation: A population of chromosomes, each representing a possible solution to the problem is generated randomly. The genes comprising the chromosomes are allocated values between the upper and lower bounds of the variable values. The initial population must offer a wide diversity of genetic materials. With a sufficiently large population size, adequate representation will be achieved.

2. **Decoding:** In real-value representation, no decoding is required. In binary representation, the strings are decoded to obtain decimal values which are then mapped between the limits of decision variables to obtain the parameter values (Table 4.1). For these parameter values (releases) the storage values are computed from the systems equation and checked against the storage constraints.

3. **Evaluation:** For each chromosome, the fitness is computed using an evaluation function, which is a link between the GA and the problem to be solved. For the four reservoir problem, fitness corresponds to the total benefit obtained over the operating horizon from hydropower and irrigation production. The fitness function also includes a penalty if the storage falls below the target level at the end of 12 operating period.

4. **Elitism:** The best chromosome of the previous generation is preserved such that it is not lost between generations. If the best chromosome of the current generation is worse than the best chromosome of the previous generation, the latter one replaces the worst chromosome of the current population.

5. **Selection:** Each chromosome has an associated value corresponding to the fitness of the solution it represents. For the problem under consideration, the total benefit obtained after the end of 12 operating periods is used as a measure of the fitness of the solution. Based upon this value, chromosomes are selected for participation in the next generation using the tournament selection scheme. This method randomly picks two or more chromosomes from the population and selects the best one into the next generation. The procedure is repeated for a same number of times as the population size. A discussion on tournament selection scheme is presented in chapter 3.

6. **Crossover:** Pairs of chromosomes are randomly selected and genetic information is exchanged between these chromosomes to produce new chromosomes. The crossover probabilities are usually set in the range of 0.5 - 0.95. One point, two point and uniform crossover has been used for this problem. Different crossover schemes are described in chapter 3.

7. Mutation: GAs work by eliminating chromosomes with poor fitness values in each generation and in doing so may lose some important genetic information. This might lead GAs to converge prematurely to a local optimum. To maintain diversity in the population, some of the genes are randomly mutated to keep the population from converging too quickly. Mutation occurs with specified probability which is usually very low (0.1 - 0.001). In binary or Gray representation, mutation simply means changing a 1 to a 0 or vice versa. Mutation probability of 0.01 implies that approximately 1 in every 100 bits is mutated. In this study, uniform mutation and modified uniform mutation have been used. Details of these mutation schemes are presented in chapter 3.

8. Replacement: A new population of chromosomes is obtained for further genetic processing after application of reproduction, crossover and mutation operators. These new chromosomes replace the chromosomes from the old population.

The GA repeats steps 2 to 8 to produce successive generations and to obtain improved solutions. The basic GA code which implements the procedure outlined above is presented in appendix B. The code is developed using Microsoft Visual C++.

4.7.2 The Penalty Function Approach

In a GA, the boundary conditions may not necessarily be satisfied as in LP. In the GA approach the release sequences are generated randomly and therefore it is possible that some release sequences may not satisfy the constraints on storages. Chromosomes failing to meet the constraints could be excluded from subsequent participation in the evolutionary process, but this may lead to useful genetic material being lost. Alternatives to exclusion are successive regeneration of chromosomes until they meet constraints, or imposing a penalty to reduce the fitness of chromosomes failing to meet constraints. The former approach disrupts the GA process. In this research, a penalty function approach has been used. The penalty function is quadratic, and is based upon the degree of constraint violation. This is particularly useful in dealing with long chromosomes representing a time series in which the constraints are violated in only a small part. Also, potentially good genetic material is allowed to contribute to subsequent generations when violations are small.

4.8 Evaluation Of Alternative GA Formulations

GAs are considered to be general purpose search procedures inspired by the process of natural evolution, and near optimum value of the objective function should be achieved after several repetitions of the generation cycle. In practice, the performance of GAs has been found to be sensitive to the GA parameters (DeJong 1975). A series of sensitivity analyses have been carried out to determine appropriate parameter settings under binary, Gray and real-valued coding, prior to evaluation of the alternative codings themselves. The tournament selection scheme has been used for all the GA formulations. Goldberg and Deb (1991) carried out a comparative analysis of various selection schemes and seem to favour the tournament selection scheme. Binary tournament selection has been used by Cieniawski et al. (1995) in their groundwater monitoring problem. Yang and Soh (1997) have also used the same approach for the structural optimisation of trusses.

4.8.1 Sensitivity to Crossover Probability

In many practical problems GA results are found to be sensitive to crossover and mutation probabilities. This is due to the fact that a particular bit appearing at a particular position on an individual string may disappear at an early stage of the run due to mutation or crossover. Due to the complexity of the problem, the now extinct bit may be precisely what is needed to achieve optimal performance at a later stage of the run. The sensitivity of results to crossover probability has been analysed and the results are presented in Figure 4.3.

Various researchers (e.g., DeJong 1975; Goldberg 1989) have suggested that good performance may generally be achieved using a high crossover probability and low mutation probability. It has been observed from initial test runs that for the four reservoir problem good results will be achieved when mutation is restricted to about one gene per chromosome on average. Since the chromosomes for the four reservoir problem are made up of 48 genes, it was decided to set the initial mutation probability at 0.02 (approximately 1/48). Sensitivity of the GA to crossover probability was examined with a mutation probability of 0.02 using a population size of 100. The tournament selection scheme with uniform crossover, and a modified uniform mutation operator was adopted for real-valued representation. Crossover probabilities in the range of 0.50 to 1.0 were considered through runs with a fixed length of 500 generations. Figure 4.3 shows the sensitivity of GA performance to crossover probability for different coding schemes. The results showed that the optimum benefit achieved was always within the neighbourhood of the known global optimum for each of the three coding schemes.

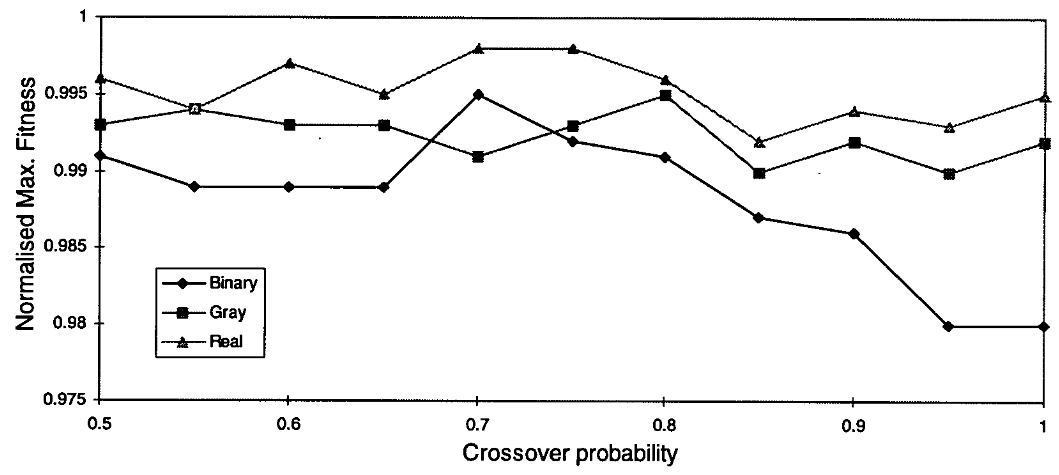


Figure 4.3 Sensitivity to crossover probability

The real-valued coding provides the best performance over a wide range of crossover probabilities. The GA was able to locate promising points in the neighbourhood of the known global optimum for a wide range of crossover probabilities. The best values were achieved for the crossover probabilities in the range of 0.7 to 0.8. It can be observed from Figure 4.3 that binary coding is apparently more sensitive to crossover probability than either the real-valued or Gray coding. The peak performance is achieved with a crossover probability of 0.7 but beyond this value, there is progressive deterioration in performance. The analysis showed that the performance of GA might not be good if too high or too low values of crossover probabilities are used.

A total of ten runs were carried out for each of the different values of crossover probabilities and in each case for the real-value and Gray coding, the best value achieved was more than 99% of the known global optimum. The moderate crossover probabilities in the range of 0.6 to 0.8 produce better results as they allow some chromosomes to undergo mutation only. This is particularly helpful in the later stages of the run when only slight changes to the existing chromosomes are required to improve them. The results demonstrate clearly that the GAs are robust with reasonable results being achieved over a fairly wide range of crossover probabilities and representation schemes. The results also demonstrate that real-value coding is more robust and produces better results than either of the binary or Gray code representations.

4.8.2 Sensitivity to Mutation Probability

The sensitivity of the objective function to mutation probability for different representation schemes was also investigated. Since the best value of objective function was achieved with a crossover probability of 0.7, the sensitivity to mutation probability was analysed using this value. The mutation probability may be expressed as

$$p_{mutate} = \frac{k}{C_{length}} \tag{4.10}$$

where p_{mutate} is the mutation probability, k is the number of mutations, and C_{length} is the chromosome length in genes. Numbers of mutations in the range of 0.1 to 10 per chromosome have been considered. Since the chromosome length for this problem is 48, mutation probabilities in the range of 0.002 to 0.208 have been used. Figure 4.4 shows the results of sensitivity analysis.

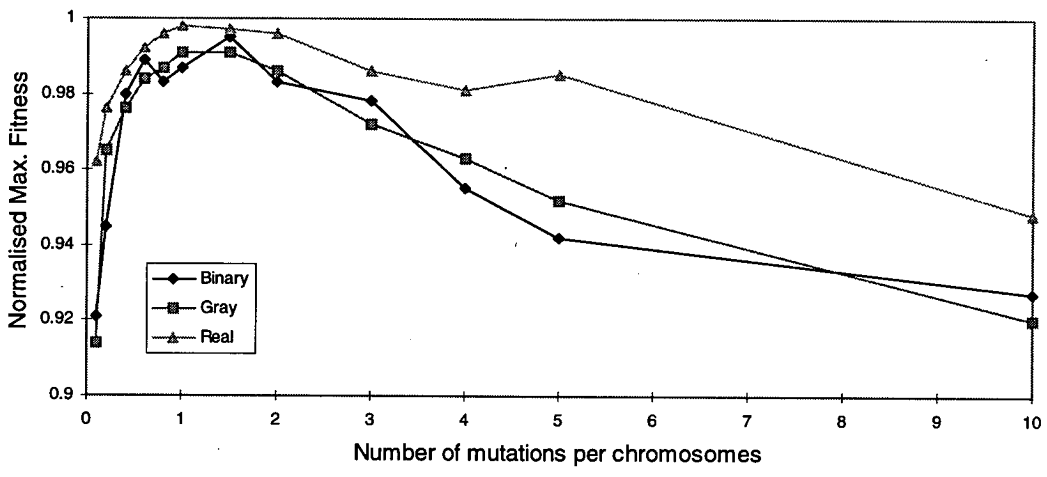


Figure 4.4 Sensitivity to mutation probability

Clearly, with real- value coding best results are achieved when k =1. In binary coding, the best result was achieved with 1.5 mutations per chromosome, but it is clear that there is more variability in binary coding than in real-value coding. If the number of mutations per chromosome is large, say more than 4, the performance is likely to be poor. Clearly, the real-valued coding outperforms the binary and Gray coding. There is again an indication of robustness, and 0.6 to 2 mutations per chromosome produces reasonable results. The results of sensitivity analysis for mutation probability showed that too high or too low a mutation probability does not produce good results. A proper choice of mutation probability is vital for good performance of a GA. It should be noted that the results are more sensitive to mutation probability than to crossover probability.

4.8.3 Analysis of Representation Schemes

The performance of GA depends upon the proper choice of GA parameters such as crossover probability and mutation probability. The results of sensitivity analysis presented in Figure 4.3 and Figure 4.4 indicate that the real-valued representation is less sensitive to the values of crossover and mutation probabilities used. It can be observed that the best values of objective functions are produced with a crossover probability of 0.7. Also, a mutation probability corresponding to one mutation per chromosome appears to be optimal. Based upon these values, an evaluation of binary, Gray and real-value representation schemes has been carried out.

The variation of best-of-generation maximum fitness with the number of generations for each of the coding schemes is shown in Figure 4.5. The trend of variation is similar for each of the schemes, although it is apparent that the real-valued representation produces a smoother curve, perhaps indicating greater robustness. The real-value representation exhibits a slower rate of improvement in fitness in the mid-generation period of the run, but sustains its rate of improvement for longer. The binary representation makes more erratic progress towards better fitness than do either the Gray code or real-value representations. All three schemes quickly approach the neighbourhood of the optimum within the first 50 generations. The real-value representation maintains a steadier rate of improvement than the other schemes. It appears that the binary and Gray code become restricted partly through mutation. Mutation may occur in any part of a gene, and while this is helpful in early generations, it can be disruptive in later ones. Since elitism is used which ensures that the best individual is never lost between generations, a high and improving normalised maximum fitness is maintained.

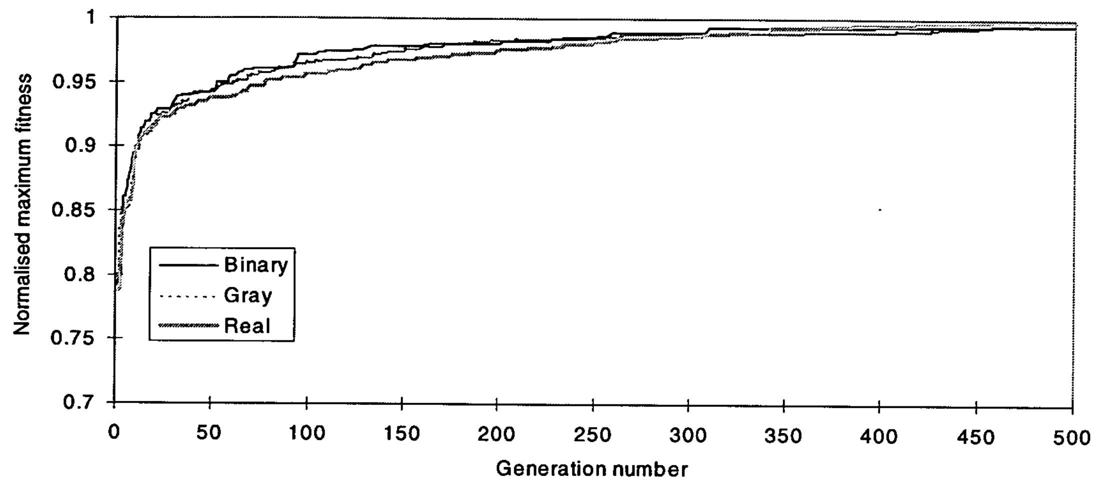


Figure 4.5 Maximum fitness with alternative representation schemes

Plots of average fitness in the population with generation number are shown in Figure 4.6. Clearly, the variability in average fitness is best for the real-value representation. This may be attributed to the improved manner in which mutation has been dealt with. The modified uniform mutation operator changes the genes by a small amount compared to the original uniform operator in which the changes could be large. With binary and Gray coding, the average fitness does not improve significantly after about 100 generations. This is partly because if an infeasible solution is generated as a result of crossover or mutation, a penalty is assigned to it and the average fitness of the population goes down.

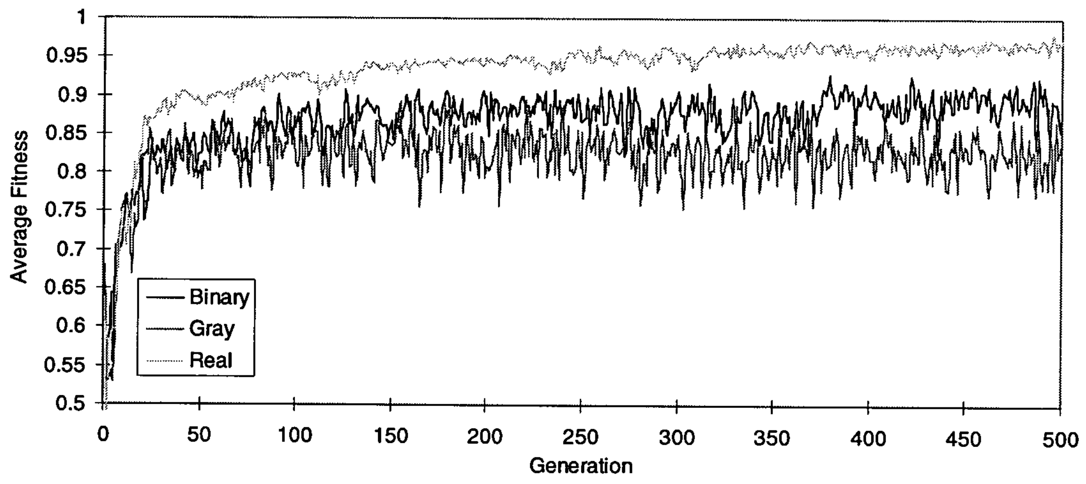


Figure 4.6 Average fitness of population with alternative representation schemes

The maximum values of the objective function achieved after 500 generations with each coding are given in Table 4.2. The known global optimum for the problem is 401.3. The values achieved by each of the coding schemes are very close to the known optimum. Real-value coding has a number of advantages over both binary and Gray coding which can be summarised as:

- a higher fitness value is achieved;
- a smoother convergence to solution is obtained;
- formulation is straightforward, no mapping is required;
- average fitness of the population generated is much higher than with binary or Gray coding;
- execution time is less.

The results of GA could be further improved by allowing the evolution process to run for some extra number of generations. It was possible to obtain the known global optimum for the four reservoir problem after 750 generations using the real-value representation. With a population of 200, the optimum was achieved in 500 generations. A comparison of execution time for each coding scheme is shown in Table 4.2. The real-value representation completed 500 generations two and a half times faster than the binary representation and four times faster than the Gray code representation. The speed advantage associated with real-value coding could be of importance for larger problems.

Table 4.2 Normalised and actual fitness values after 500 generations

Coding	**Normalised Fitness**	**Actual fitness**	**Execution Time (seconds)**
Binary	0.995	399.0	120
Gray	0.995	399.0	175
Real-value	0.998	400.5	45

4.8.4 Influence of Population Size

Consideration has been given to the influence of population size on the performance of GA. The real-value representation was adopted, and the GA was run with different population sizes for 500 generations. The sensitivity of achieved maximum fitness to population size is shown in Figure 4.7. Acceptable results are produced with a population of 100, but by increasing the population size to 200, it was possible to reproduce the known global optimum. A decrease in population size decreases the total number of evaluations performed, and hence desired results may not be achieved with smaller population sizes. Moreover, smaller population size cannot maintain diversity in the population. The choice of proper population size depends upon the judgement and experience of the user.

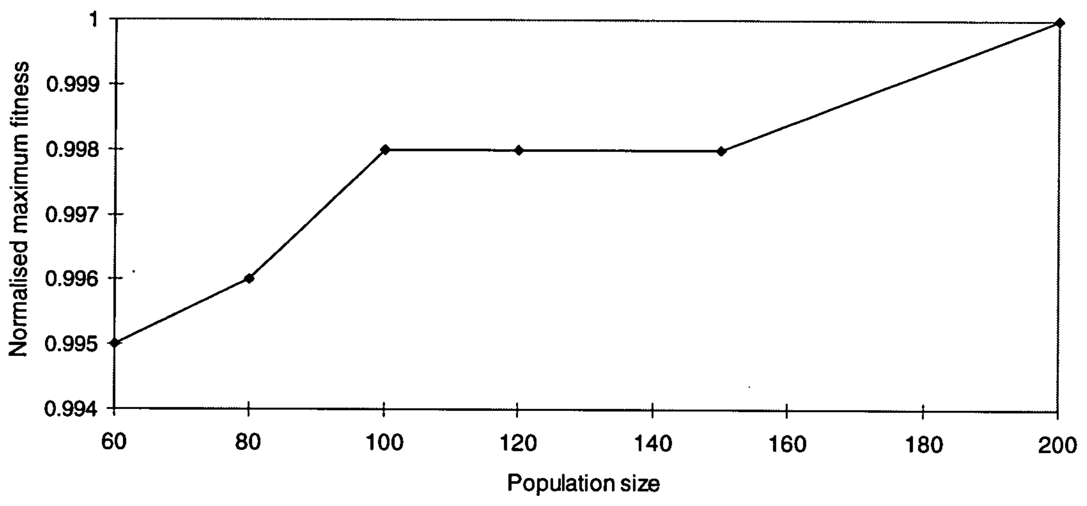


Figure 4.7 Influence of population size on fitness

4.8.5 Influence of Crossover and Mutation Schemes

The influence of crossover and mutation schemes on the performance of GA has also been analysed. Table 4.3 presents the results of analysis. Uniform crossover appears to be the best crossover operator although one point and two point crossover give similar performance. For this particular problem, the effect of crossover schemes is found to be insignificant although slightly better results are achieved using uniform crossover. This might be expected as uniform crossover is likely to lead to greater diversity within the population than either the one point or two point crossover.

Mutation seems to effect the GA process significantly. Uniform mutation operates by replacing the selected gene value by another gene value from a pool of all possible values. This may, at times cause large changes to the existing genes which may lead to loss of good genetic material. On the other hand, the modified uniform mutation operator causes fixed mutations rather than random mutations to the gene values. This dampens the impacts of mutation, and leads to better final solutions as the disruption to the solution is less.

Table 4.3 Evaluation of crossover and mutation schemes

Scheme Adopted	Normalised Maximum Fitness
Crossover Schemes	
Uniform Crossover	0.998
One Point Crossover	0.997
Two Point Crossover	0.994
Mutation Schemes	
Uniform Mutation	0.983
Modified Uniform Mutation	0.998

4.8.6 Storage Trajectories

Comparisons of the state trajectories produced by LP, DDDP, and by real-valued GA are presented in Figure 4.8 to Figure 4.11. The GA results are based on the best parameter set resulting from sensitivity analysis, and are given for a population size of 100 at the end of 500 generations. For reservoir 1, and 4, there is an exact reproduction of the optimal trajectories. There are minor shifts in trajectories for reservoir 2 and 3. The trajectories produced after running the GA process for 750 generations exactly matched the optimal trajectories obtained by LP and DDDP. It can be observed from the trajectories that for each reservoir the GA was able to find solutions which satisfied the boundary conditions.

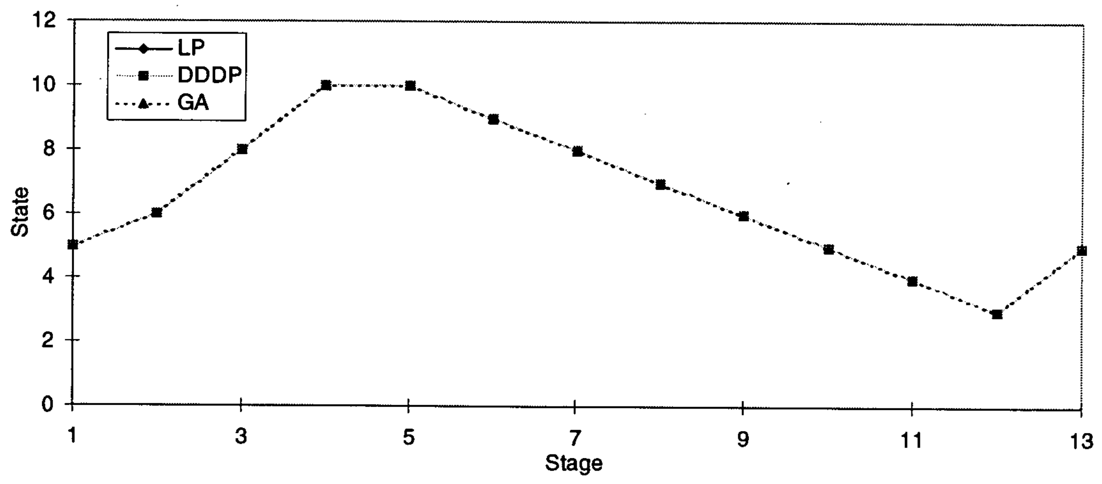


Figure 4.8 State trajectories for reservoir 1

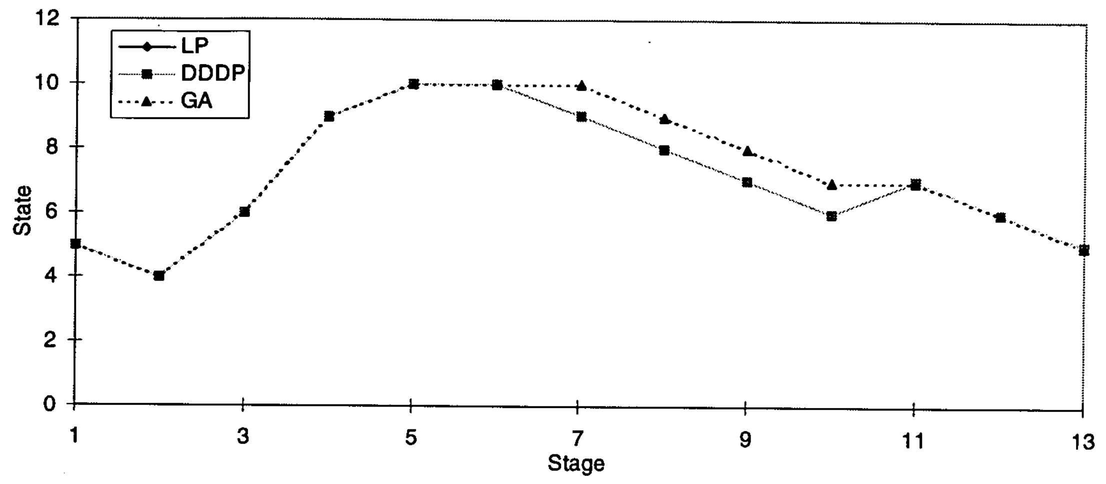


Figure 4.9 State trajectories for reservoir 2

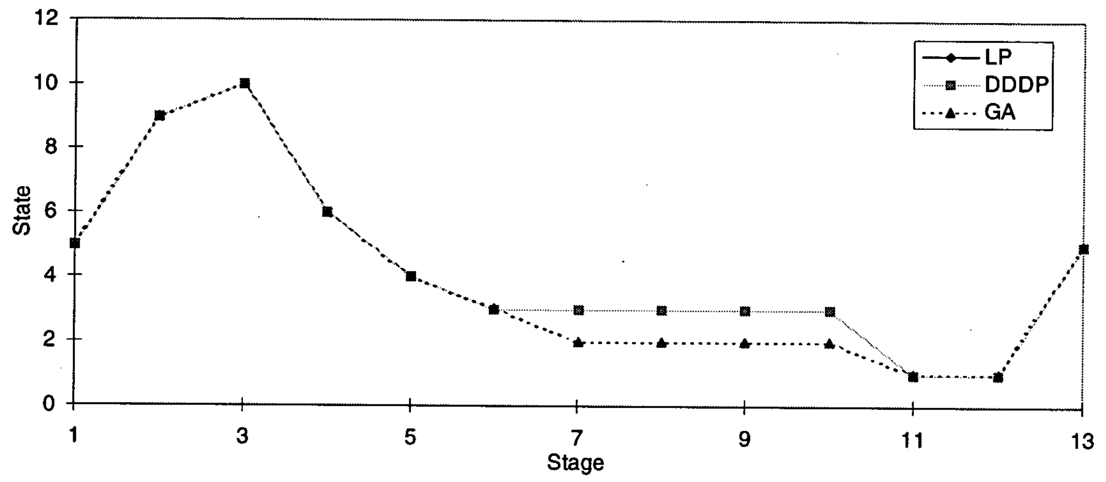


Figure 4.10 State trajectories for reservoir 3

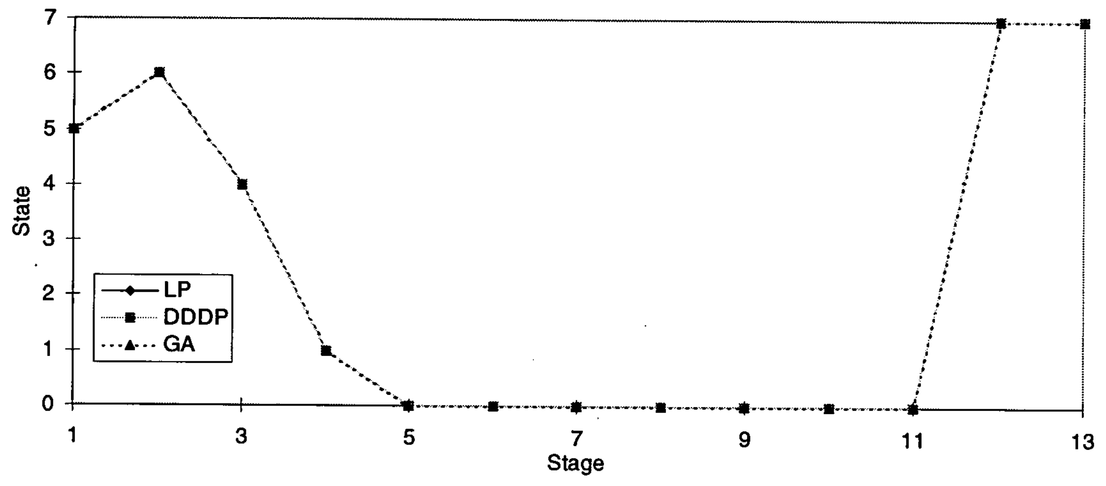


Figure 4.11 State trajectories for reservoir 4

The GA was also run with different seeds in the stochastic generators than had been used in the sensitivity runs. In 5 out of 10 runs with 750 generations, the optimal solution was produced. The values produced by other runs were only marginally lower than the optimum (0.999 for three of them and 0.998 for the other two). It can therefore be concluded that the results are relatively stable with respect to particular random number sequences.

4.9 Applications to Extended Time Horizons

As the number of time steps in the operating horizon increases, the length of chromosomes increases, and it may become difficult to achieve solutions that satisfy constraints throughout the length of chromosomes. This potential limitation has been investigated by extending the four reservoir problem to incorporate up to 96 stages instead of the original 12. Results for the problem with extended time horizons produced with the GA have been compared with those produced by DDDP, and are presented in Table 4.4.

Table 4.4 GA performance with extended time horizons

Stages	DDDP	GA with a Population of 100			GA with a Population of 200		
	Return	Return	% of DDDP	Gen.	Return	% of DDDP	Gen.
12	401.3	400.5	99.80	500	401.3	100	500
24	810.6	808.7	99.70	1300	808.9	99.79	900
36	1220.1	1219.1	99.92	2400	1218.6	99.89	1900
48	1629.6	1621.2	99.48	2600	1626.5	99.81	2100
72	2448.6	2439.2	99.62	4200	2446.0	99.89	4100
84	2858.1	2842.5	99.45	4700	2847.5	99.63	4400
96	3267.6	3253.3	99.56	5300	3259.8	99.76	5500

Note : Gen. stands for the number of generations required to reach the convergence condition.

With longer chromosomes, a larger population size should be used in order to maintain diversity in the population. It is clear from Table 4.4 that the GA maintains high fitness, and that the deterioration in fitness at longer time horizons is not significant. The number of generations included in Table 4.4 are those required for GA to converge to a state at which the change in fitness over 100 generations is less than 0.1 (0.025%). The results indicate that a larger population does produce better overall fitness, although interestingly, the number of generations required with a larger population is very similar to the number required with a smaller population.

4.10 The Modified Four Reservoir Problem

The application of GAs to the four reservoir problem has demonstrated their ability to find optimal or near optimal solutions. GAs have considerable flexibility in application to non-linear problems. To demonstrate this, the four reservoir problem has been modified to include a non-linear objective function and has been solved by GA. The hydropower generated from each of the reservoirs was assumed to be a non-linear function of storage and release from the reservoir. The head-storage relationship for each of the reservoirs was assumed to be of the following form.

$$H = KS^n \tag{4.11}$$

where H is the water level corresponding to storage S in standardised units and K and n are constants. The values of K and n used for this problem are 3 and 0.4 respectively. Figure 4.12 shows the relationship between water head and storage.

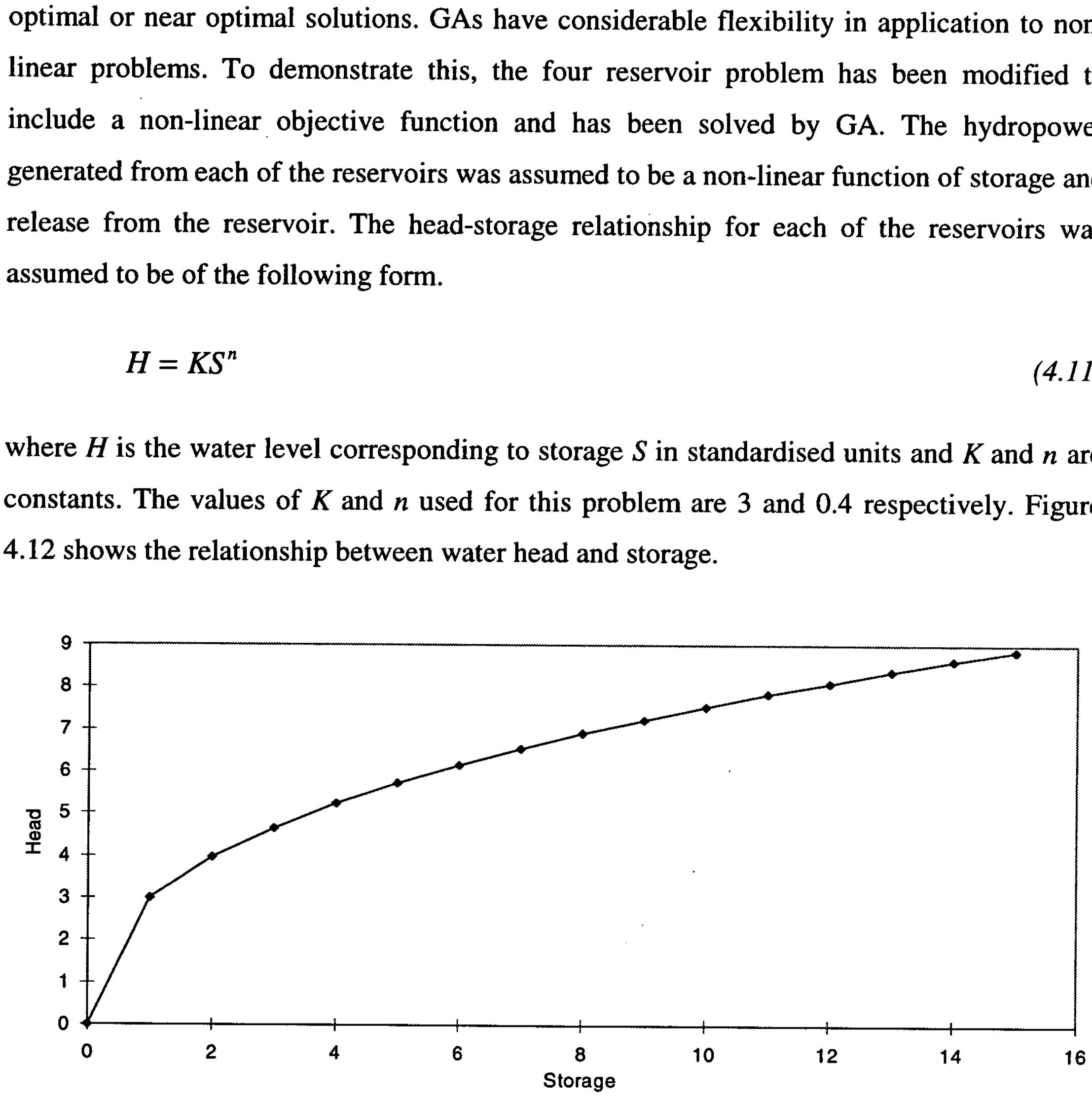


Figure 4.12 Elevation versus storage relationship for the modified four reservoir problem

The objective is to maximise the economic returns from the production of hydropower from the reservoirs subject to the constraints described for the four reservoir problem. Mathematically, the objective function for the modified problem could be expressed as:

$$\textit{Maximise} \quad F=\sum_{i=1}^{4}\sum_{t=1}^{12} b_i(t).\,p_i(t)+\sum_{i=1}^{4} g_i\left[s_i(13),\ d_i\right] \tag{4.12}$$

where $p_i(t)=f\,[s_i(t),\ u_i(t)]$ is the power generated from reservoir i in time step t. This is a non-linear function due to the non-linear relationship between the water level and the storage given by (4.11). The constraints on storages, releases, and initial and target storages are same as that for the original problem.

4.10.1 Solution Procedure and Results

The original problem formulated by Larson (1968) is solvable by LP. The modified four reservoir problem has been solved using DDDP and using the GA technique. The DDDP procedure does not require penalty functions to satisfy the terminal state conditions of the problem because it is possible to trace back those trajectories that have the required final state. However, a penalty function is required in the GA approach in order to satisfy the terminal conditions. For the GA, the penalty function used with the linear four reservoir problem has been adopted. The GA was set up with real value representation, tournament selection, elitism, uniform crossover and modified uniform mutation. The same procedure as used for the original four reservoir problem has been used for this problem to obtain a solution by the GA technique. The problem has been solved retaining a population size of 100. The same GA computer code as used for the original four reservoir problem has been used and only the evaluation function needed to be modified to incorporate the non-linear power function.

With the parameters adopted, the optimal return for the problem found using DDDP was 1901.39. The DDDP procedure was started with an initial trial state trajectory having a return of 1475.90. After 10 iterations, there was no further improvement in the value of objective function, and the DDDP procedure terminates. Figure 4.13 shows the computational results for the DDDP technique. The GA was able to produce the same optimum within 500 generations. As expected the non-linearity of the objective function does not effect the performance of GA in any way. Figure 4.14 shows the manner in which the GA approached the solution for a population size of 100, a probability of crossover of 0.70 and a probability of mutation of 0.02. These parameters have been chosen based upon the results of original four reservoir problem. It can be observed from Figure 4.14 that the GA quickly approached the optimum for this problem, after about 450 generations. The average fitness of the population was very high indicating that a large number of alternative solutions close to the optimum have been produced.

Initial trial trajectories for the DDDP and the optimal trajectories obtained using DDDP and GA are shown in Figure 4.15 to Figure 4.18. The state trajectories produced by GA match exactly to those produced by DDDP. The GA executed almost three times faster than the DDDP, and required no initial trial trajectories to be set. The execution time for GA was 45 seconds compared to 125 seconds for the DDDP.

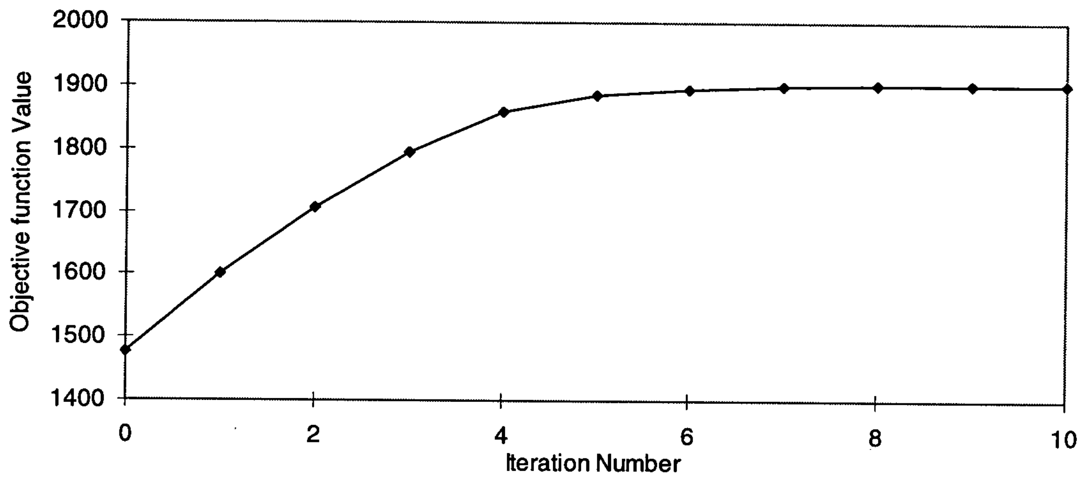


Figure 4.13 Computational results using DDDP for the modified four reservoir problem

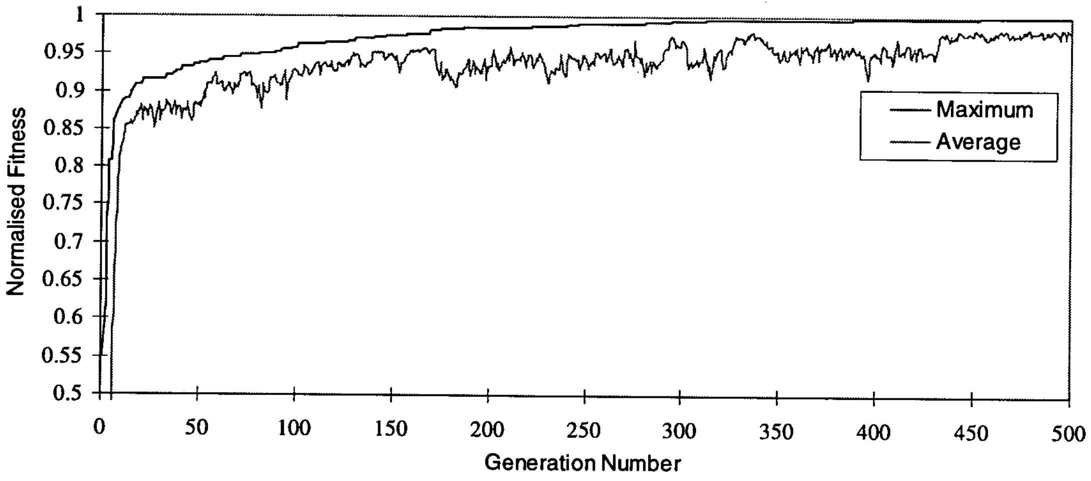


Figure 4.14 Fitness variation for the modified four reservoir problem

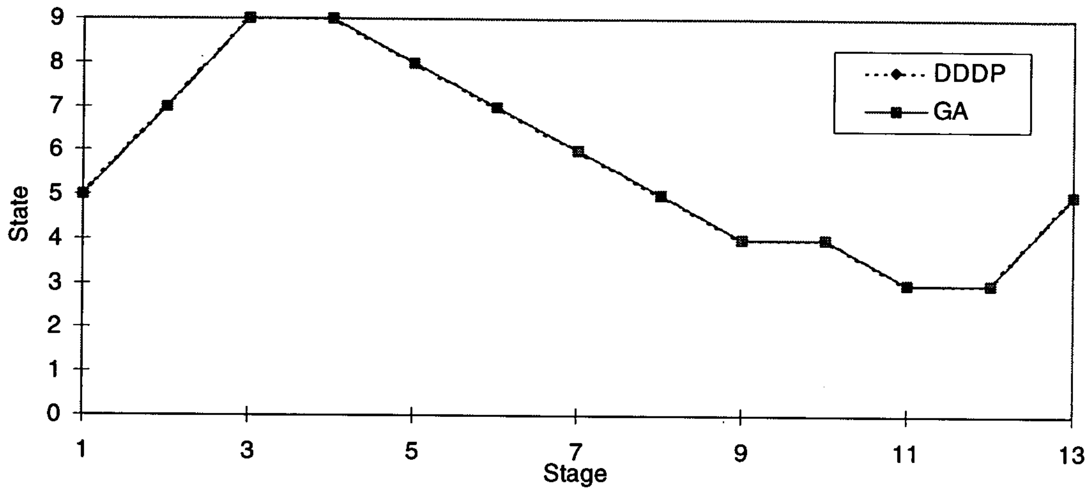


Figure 4.15 State trajectories for reservoir 1, modified four reservoir problem

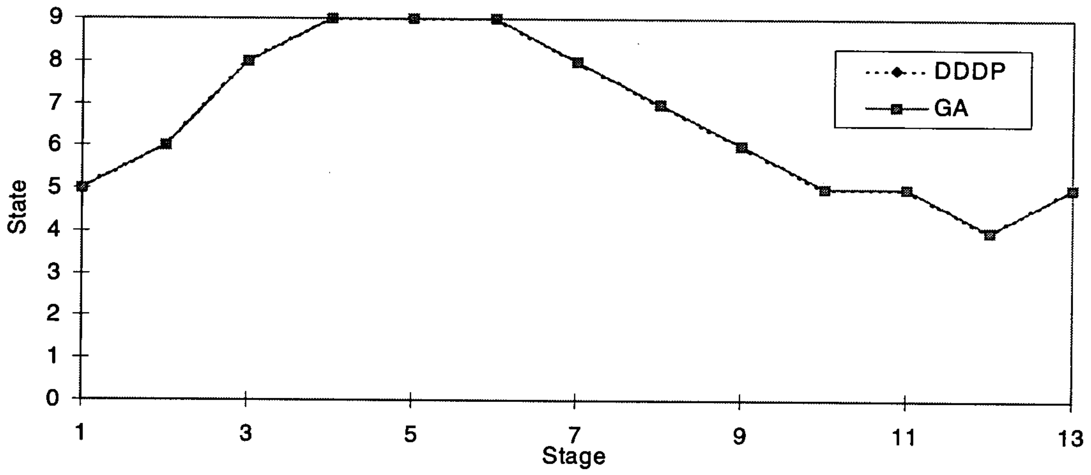


Figure 4.16 State trajectories for reservoir 2, modified four reservoir problem

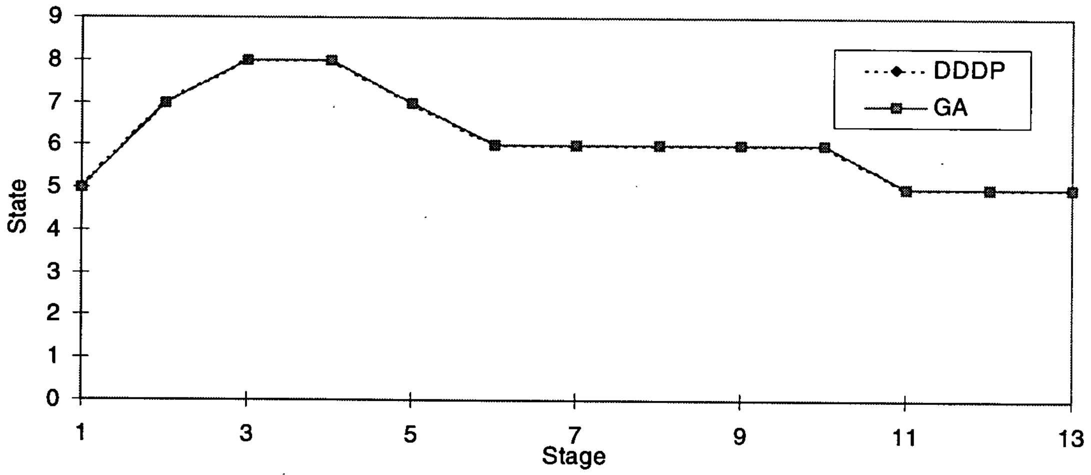


Figure 4.17 State trajectories for reservoir 3, modified four reservoir problem

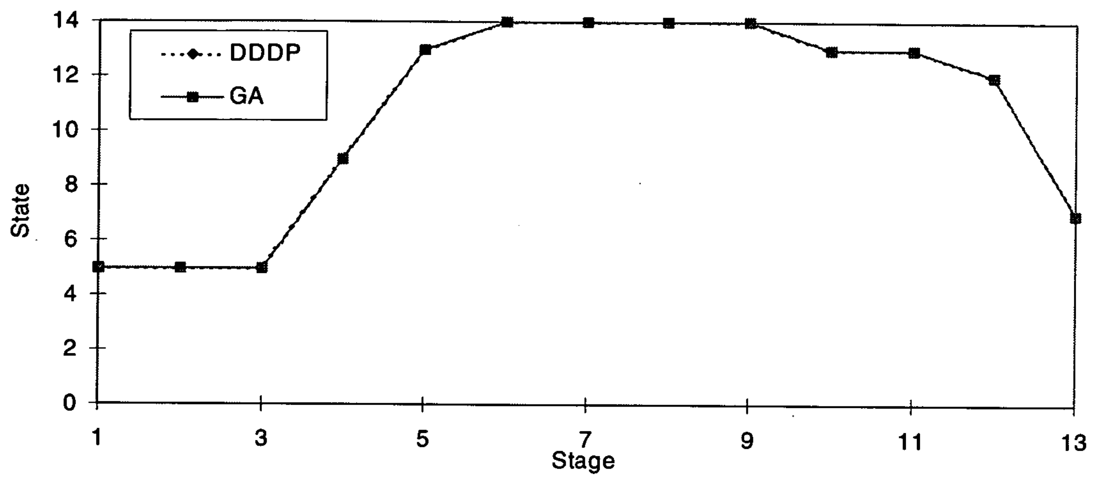


Figure 4.18 State trajectories for reservoir 4, modified four reservoir problem

4.11 The Ten Reservoir Problem

A ten reservoir problem was formulated and introduced to the literature by Murray and Yakowitz (1979). The problem is complicated not only in terms of size, but also because of many time-dependent constraints on storage. Murray and Yakowitz (1979) presented a solution to the problem using constrained DDP. The problem is beyond the capacity of DP and it is even difficult with variants such as DDDP. The problem was formulated such that it remained solvable by LP. This is particularly useful for evaluating the results obtained by other optimisation techniques.

The schematic of the ten-reservoir problem is shown in Figure 4.19. The system comprises of reservoirs in series and parallel. The releases from the upstream reservoirs are passed on to the downstream reservoirs, and a reservoir may receive supplies from one or more upstream reservoirs. The releases from the reservoirs are used for generating hydropower. Operation of the system is to be optimised over 12 operating periods to maximise the returns from hydropower production. The benefit function is a linear function of release and is based upon the numerical values provided by Murray and Yakowitz (1979).

The decision variables for the problem are the releases from the resevoirs and the state variables are represented by the storages contained in the reservoirs i =1, 10. The upper and lower bounds on storages are defined for each reservoir and for each operating period. The inflows have been defined for the entire time series for the upstream reservoirs 1,2,3,5,6 and 8. The storage constraints, benefit functions and the inflows have been tabulated by Murray and Yakowitz (1979). The initial and target storages are same for all the reservoirs, and are given in Table 4.5. The minimum allowable storages and the constraints on the release are also defined for each reservoir and are shown in Table 4.5. Details of the problem can also be found in the work by Murray and Yakowitz (1979).

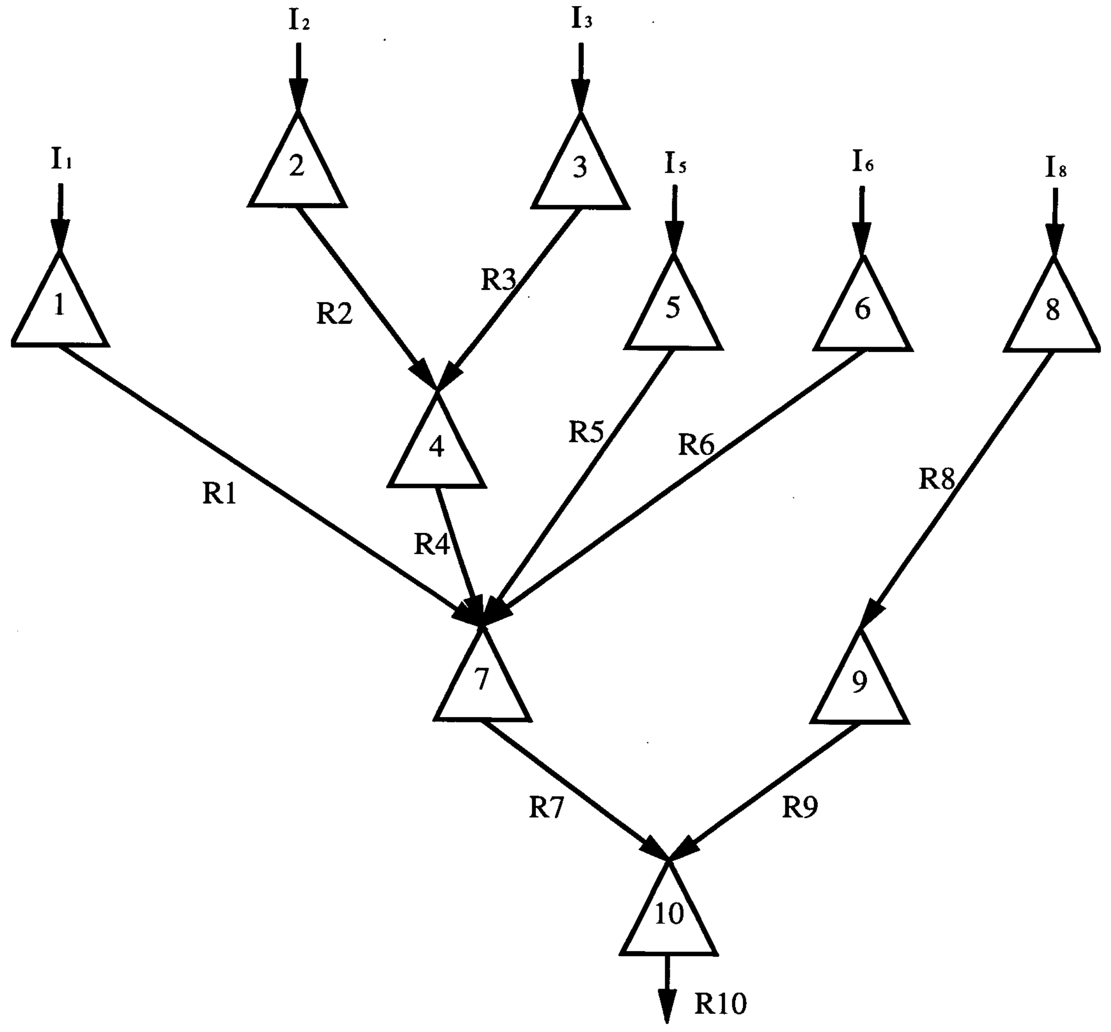


Figure 4.19 The ten reservoir system

Table 4.5 Bounds on releases and storage

Reservoir	**Release**		**Storage**		
	Minimum	**Maximum**	**Minimum**	**Initial**	**Target**
1	0.005	4.0	1	6	6
2	0.005	4.5	1	6	6
3	0.005	2.12	0.3	3	3
4	0.005	7.0	1	8	8
5	0.006	6.43	1	8	8
6	0.006	4.21	1	7	7
7	0.01	17.1	1	15	15
8	0.008	3.1	1	6	6
9	0.008	4.2	0.5	5	5
10	0.01	18.9	1	15	15

Note: These constraints apply in all time-steps.

The dynamic behaviour of the system at any stage t = 1, 12 is described by the following continuity equation.

$$S_i(t+1) = S_i(t) + I_i(t) + MR_i(t) \tag{4.13}$$

where $\mathbf{S_i(t)}$ = vector of reservoir storages, $\mathbf{I_i(t)}$ = vector of reservoir inflows, $\mathbf{R_i(t)}$ = vector of reservoir releases in time period $t=1, N$ from reservoirs $i=1,n$; and **M** = n x n matrix of indices of reservoir connections, N is the number of time steps considered, and n is the number reservoir included in the model.

The matrix **M** is a n order square matrix with -1 along the diagonals and +1 in the position ith column and jth row if the release from reservoir i goes into reservoir j. The rest of the martrix elements are zero. From this consideration, the matrix **M** could be expressed as follows.

$$M = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{bmatrix} \tag{4.14}$$

The performance criterion to be maximised is the sum of returns due to power generated by the reservoirs. The performance criterion also includes a penalty if the storage in a reservoir falls below its target level at the end of N operating periods. The penalty function was assumed to be of the following form:

$$g_i[s_i(13), d_i] = \begin{Bmatrix} -60[s_i(13) - d_i]^2 & \textit{if } s_i(13) \le d_i \\ 0 & \textit{if } s_i(13) > d_i \end{Bmatrix} \tag{4.15}$$

The penalty function is not required in LP since the boundary conditions can be set up as constraints but it is needed in the GA approach. It was found that with the ten reservoir problem, a more severe penalty function was required than with the four reservoir problem. Mathematically, the objective function can be expressed as:

$$\textit{Maximise} \quad F = \sum_{i=1}^{10} \sum_{t=1}^{12} b_i(t).R_i(t) + \sum_{i=1}^{10} g_i[s_i(13) - d_i] \tag{4.16}$$

where F is the total benefit obtained from the system after 12 operating periods and, $b_i(t)$ is the unit return from reservoir $i = 1, 10$ in time period t.

4.11.1 Solution Procedure and Results

Unlike the four reservoir problem, the components of solution vectors for the ten reservoir problem are non-integer continuous values. This implies that the genes representing the releases can take any possible value between the defined upper and lower limits. The important advantage of GAs is that although the decision variables are real values, no discretization is required when real-value representation is used. For the particular case of four reservoir problem, the genes were constrained to take on integer values only by using a random number generator which returned only integer values. This was the requirement for the four reservoir problem. For the ten reservoir problem, the random number generator was easily modified to return real values between the predefined lower and upper limits thus ensuring that the decision variables are assigned continuous variables and the whole search space is covered. The number of genes in the chromosomes depends upon the number of reservoirs and the time-steps considered in the model. For the ten reservoir problem with 12 time steps, there are 120 decision variables to be represented in a GA. Each chromosome representing a potential solution to the problem thus consists of 120 genes compared to 48 genes for the four reservoir problem. The ten eservoir problem has been solved using the same GA code as used for the four reservoir problem, with modification only to the evaluation function and the data file.

A higher population size of 500 has been used so that the desired level of genetic diversity could be maintained for initial and subsequent populations. Based on the results of sensitivity analysis carried out for the four reservoir problem, a crossover probability of 0.7 was chosen. Initial test runs were carried out to determine the best value of mutation probability. The mutation probabilities corresponding to around 1 mutation per chromosome were considered. The best value was achieved using a mutation probability of 0.005. The GA was allowed to run for 2500 generations and took 25 minutes to run on a Pentium based PC. The variation of maximum and average fitness with the generation number is shown in Figure 4.20. It can be observed from the plot that the GA continuously improves the solution although it appears that the rate of improvement is slow during the later part of the run. The variability in average fitness over generations is low and during the end of the run, average fitness of the population approaches the maximum fitness. This shows that the population has matured and that very little improvement in the fitness values could be obtained even if the algorithm is run for many more generations.

The optimal return for the problem using LP is 1194. Murray and Yakowitz (1979) obtained a value of 1190.652 using constrained DDP. The value obtained by GA approach is 1190.25, which is around 99.7% of the known global optimum for the problem. Given the size and complexity of the problem, the results of GA are highly satisfactory. The trajectories produced by GA matched closely with those produced by LP (Figure 4.21 to Figure 4.30). The execution time for the GA was 8 times longer than that of the LP on a Pentium based PC. Introduction of non-linear objective function or constraints would not, however, have influenced the execution time of the GA. This is an advantage for complex systems. The purpose of this application has been to demonstrate that GAs are capable of addressing large problems. Further complexity introduced through non-linearities in any part of the system would not present any difficulty for the GA approach, and this is a particular strength of the approach.

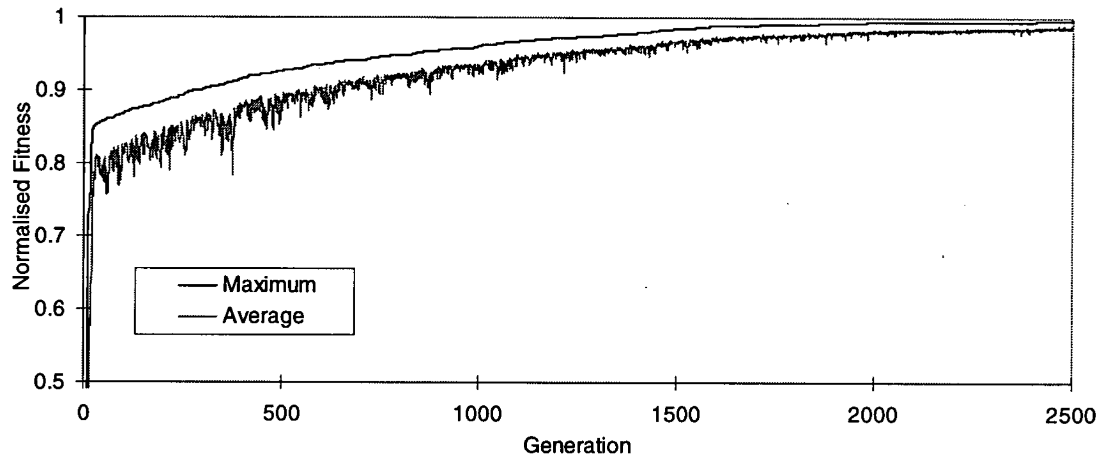


Figure 4.20 Generation versus fitness for the ten reservoir problem

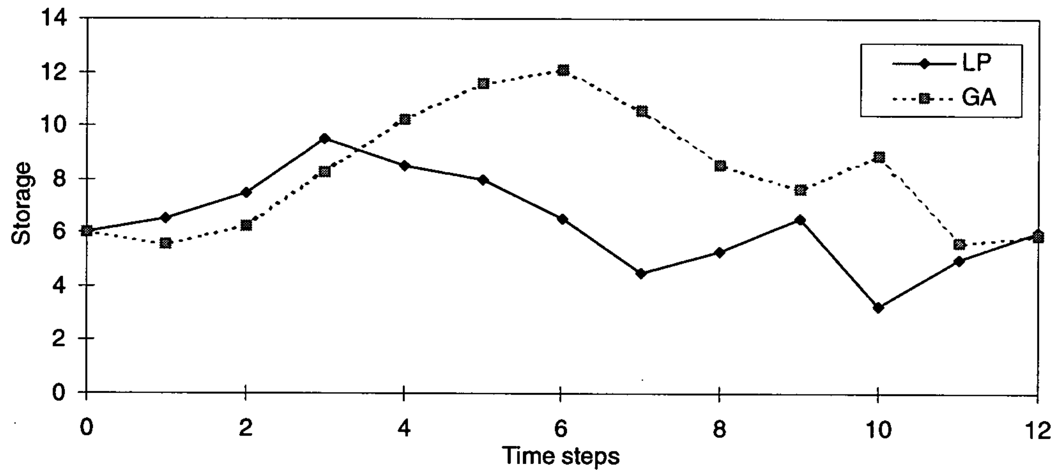


Figure 4.21 Storage trajectories for reservoir 1

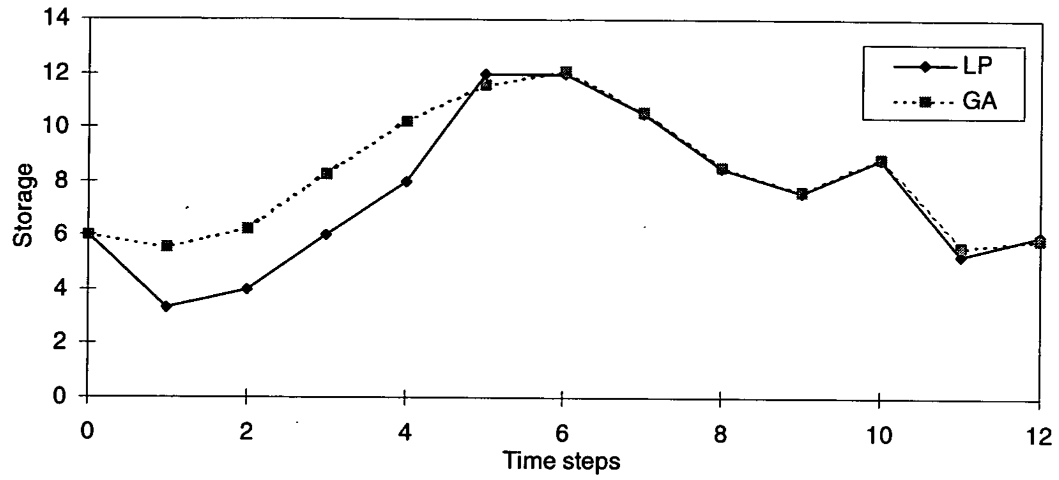


Figure 4.22 Storage trajectories for reservoir 2

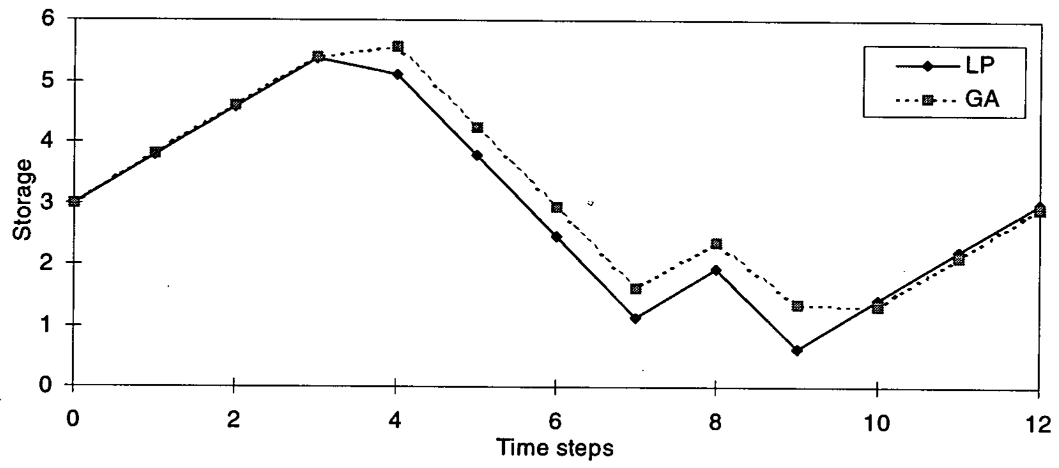


Figure 4.23 Storage trajectories for reservoir 3

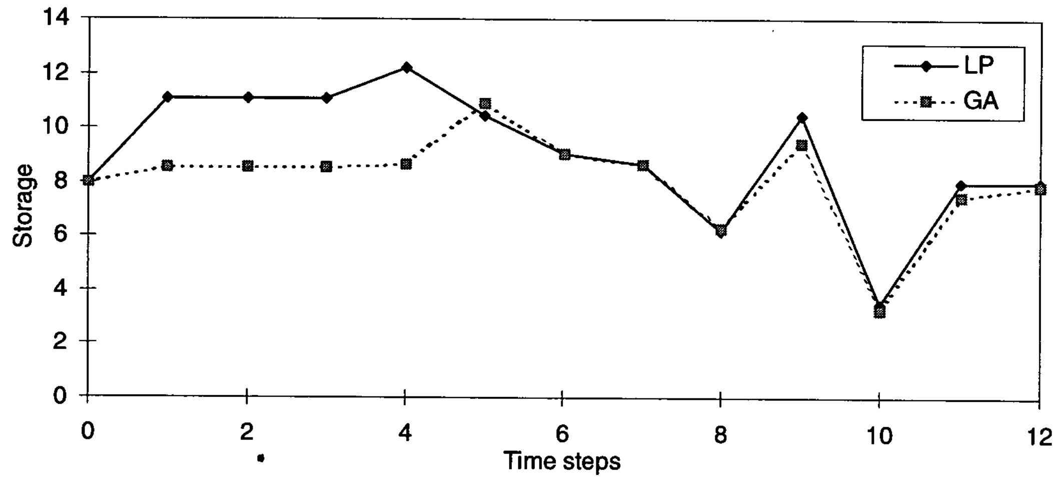


Figure 4.24 Storage trajectories for reservoir 4

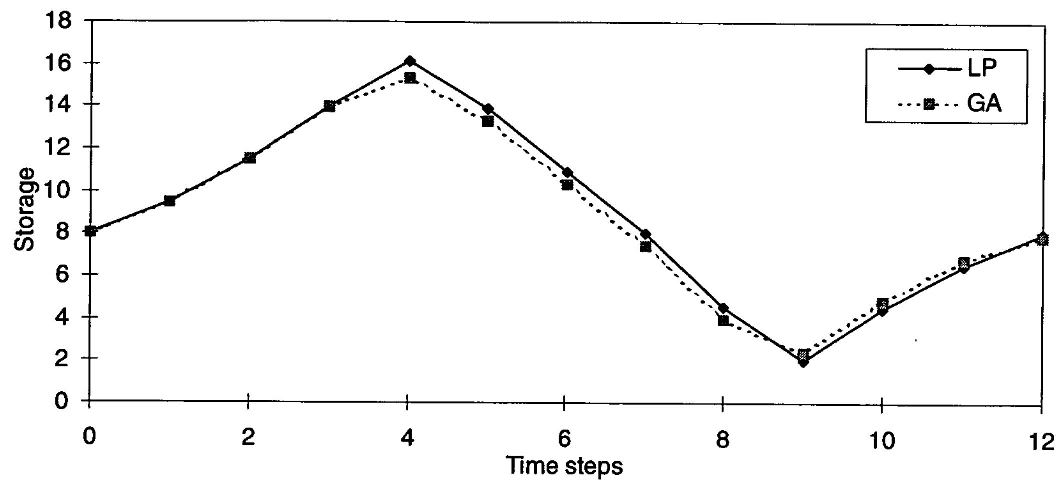


Figure 4.25 Storage trajectories for reservoir 5

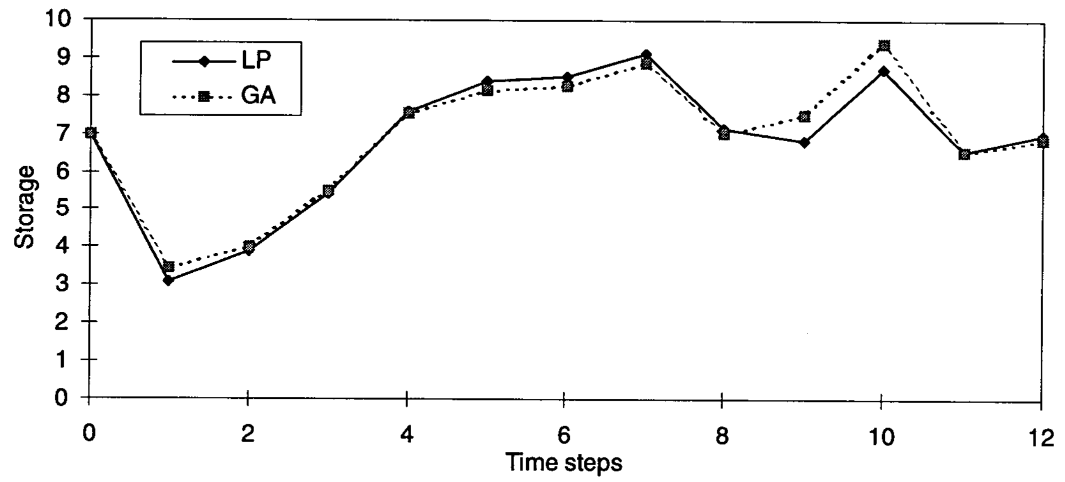


Figure 4.26 Storage trajectories for reservoir 6

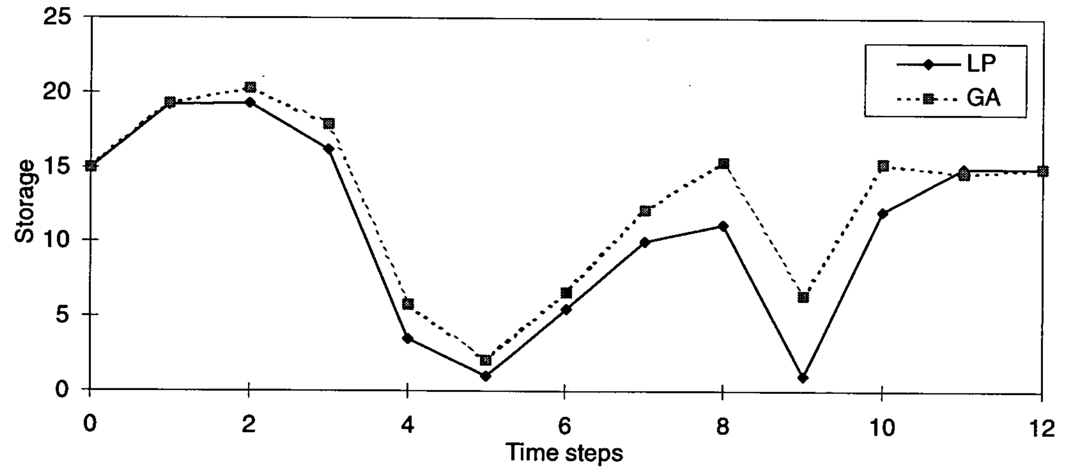


Figure 4.27 Storage trajectories for reservoir 7

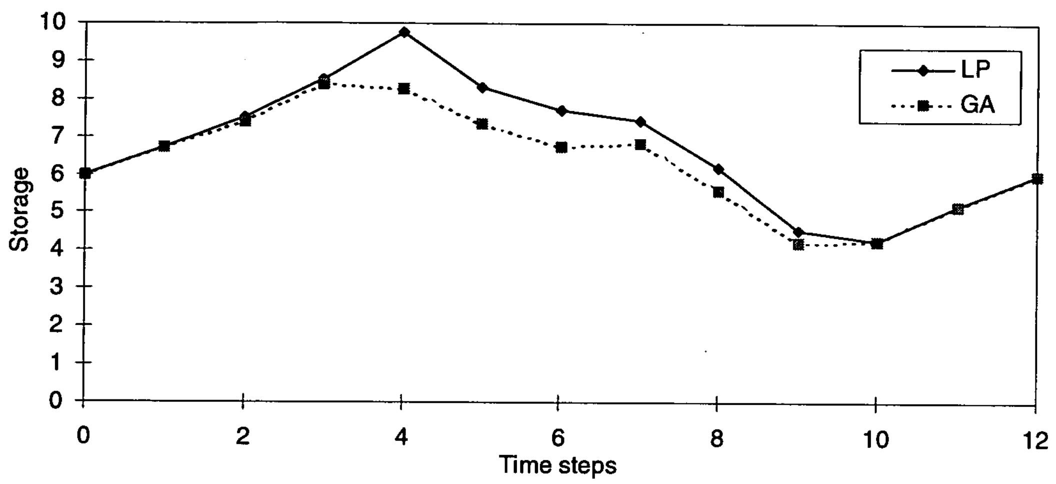


Figure 4.28 Storage trajectories for reservoir 8

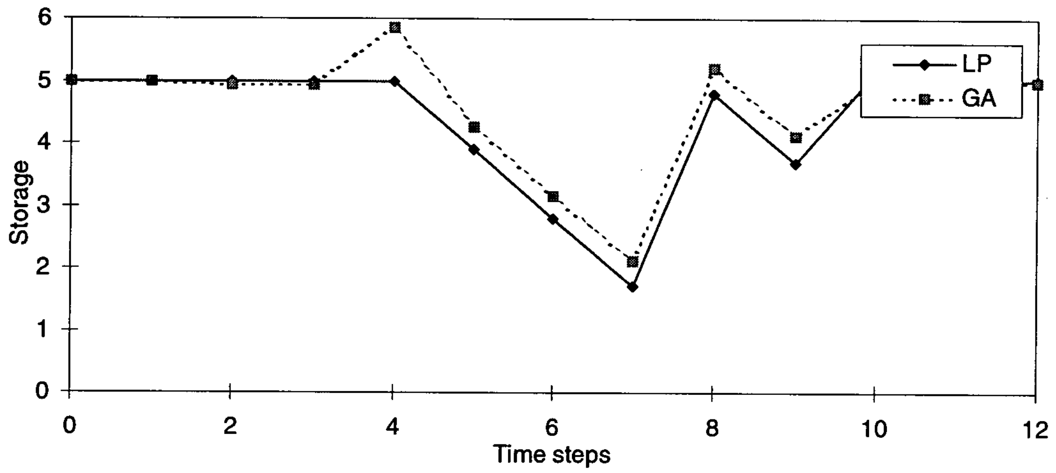


Figure 4.29 Storage trajectories for reservoir 9

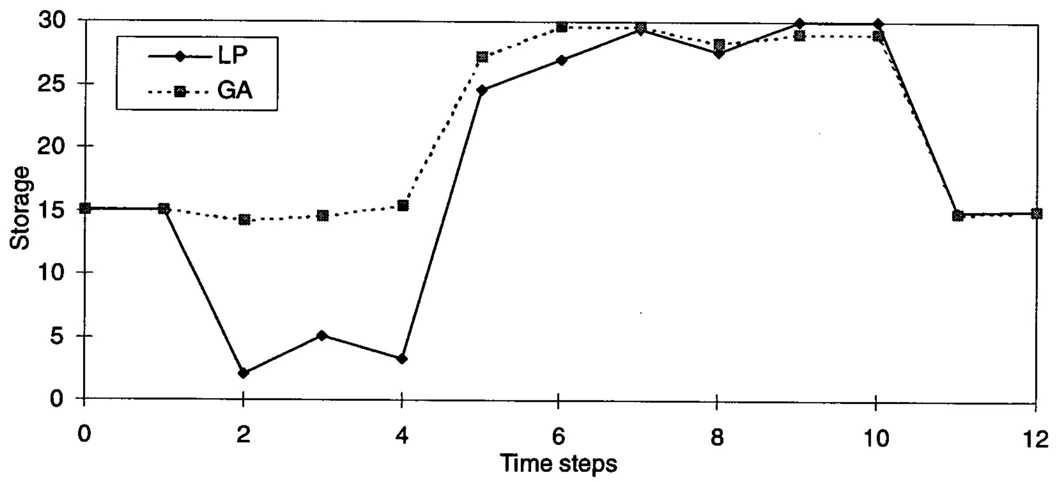


Figure 4.30 Storage trajectories for reservoir 10

4.12 Conclusions

The four reservoir problem has been solved satisfactorily using the GA approach. Several possible formulations have been considered along with their sensitivity to various parameters. It is concluded that for the four reservoir problem solved here, a real-value representation incorporating tournament selection, elitism, uniform crossover and modified uniform mutation will operate most efficiently and will produce the best results. Sensitivity analysis on GA parameters showed that a crossover probability of 0.70 and a mutation probability corresponding to one mutation per chromosome is most appropriate. A solution very close to the known global optimum can be achieved within 500 generations with a population of 100. With a population of 200, the known global optimum can be achieved in 500 generations, and with a population of 100, the known global optimum can be achieved in 750 generations.

Application of the approach to the modified four reservoir problem demonstrated that GAs can easily be applied to non-linear problems. It has also been demonstrated that large problems such as the ten reservoir problem could also be solved using the developed GA approach. It has been shown that acceptable results can be produced over longer time horizons. Furthermore, considerable economy in execution time can be achieved by performing the function evaluations in a parallel fashion since these evaluations are not interdependent. GAs are by nature highly parallel but have traditionally been run on serial machines only because of the limited availability of parallel computers.

The results presented in this chapter indicate that there is potential for application of GAs to large finite-horizon multi reservoir systems problems, as well as to stochastic optimisation problems. A significant advantage of the GA approach is that no initial trial release policy is required as in DDDP, for example. The approach is easily applied to non-linear problems and to complex systems. Another significant advantage of GA is that it produces a range of feasible solutions within the neighbourhood of the optimal solution, thereby providing more choice and flexibility to the decision maker. The results indicate the robustness of the search method that underlies the GA approach and the flexibility of the formulation itself.

5. THE BRANTAS BASIN

5.1 Introduction

The practicality of the developed GA approach is investigated through application to a real reservoir system in Indonesia. In this chapter, the characteristics of the Brantas Basin are described. The details of the simulation model of the basin are also presented. The development of water resources in the basin dates back to middle of 19th century with the construction of facilities for supplemental wet season irrigation. The basic system of the existing irrigation network was completed in the 1930's. Mott MacDonald (1981) carried out a comprehensive water resource availability study for Brantas Basin. The study showed that water resources in the basin are fast approaching their limits, but the demand for potable water is increasing rapidly. It is therefore important that the available water resources are managed in the best possible manner.

Following this introduction, section 5.2 describes the characteristics of the basin and those of storage reservoirs. The previous studies carried out in the basin are also discussed in the same section. Section 5.3 discusses the existing water use in the basin. The structure of an existing simulation model is described in section 5.4. Results of some simulation model runs are presented in section 5.5. The conclusions are presented in section 5.6.

5.2 The Basin

This section describes the characteristics of the Brantas Basin in Indonesia and is based on the reports of Mott MacDonald (1988). The Brantas Basin is located in East Java, the largest province of Java, Indonesia. Figure 5.1 shows the main features of the Brantas Basin. The K. Brantas is the second largest river in Java and has a total catchment area of 12000 km^2 and a main stream length of 341 km. Almost half of the total basin area is under cultivation. The main stream of the Brantas irrigates 81,000 hectares (ha) of high intensity agriculture whereas the total land irrigated in the basin is about 200,000 ha. The K. Brantas forms a spiral around the G. Batuk, G. Kelud and G. Arjuno: G. Kelud is an active volcano. Finally, the K. Brantas flows into the Indonesian Ocean south of the City of Surabaya.

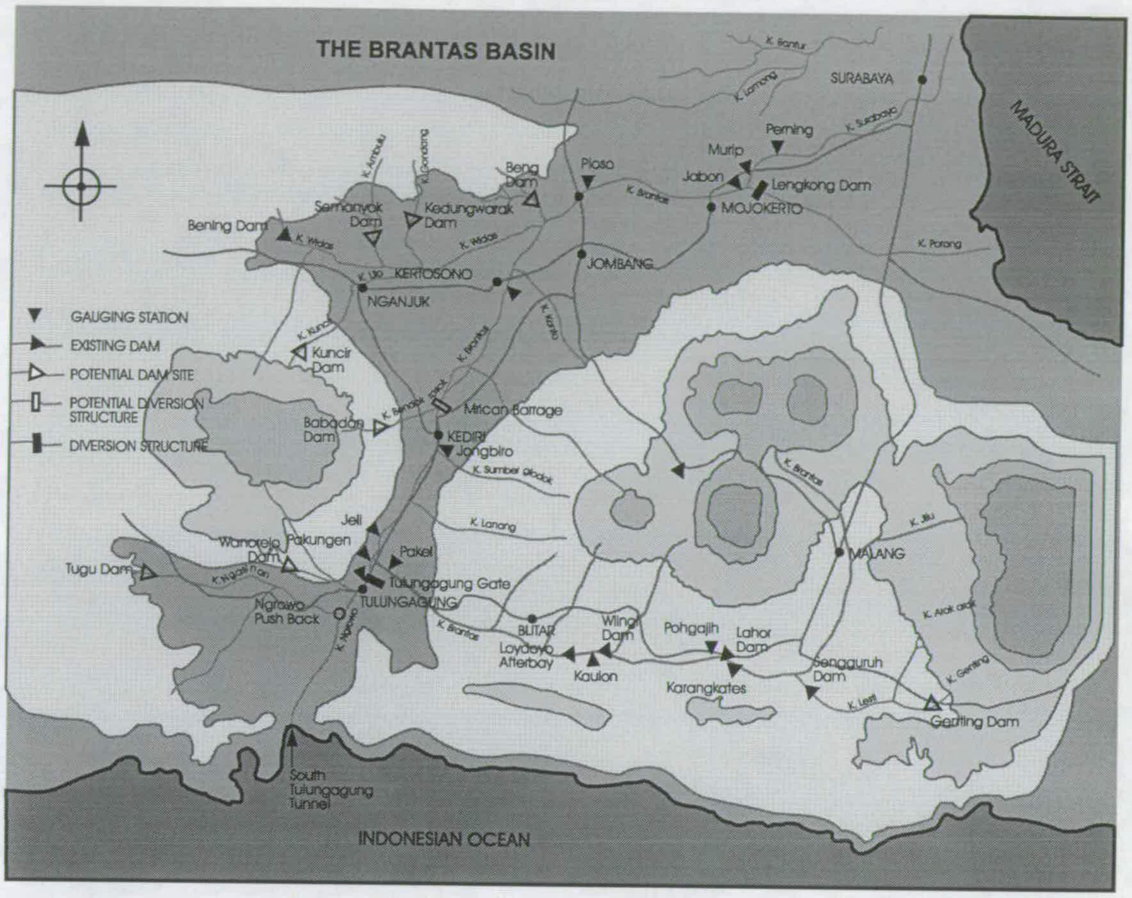


Figure 5.1 The schematic of the Brantas Basin

5.2.1 Previous Studies

The first major study of water resources in the basin was carried out by Nippon Koei (1961). The study focused largely on flood control, but came up with an overall development plan for the basin. The 1961 overall plan was reviewed by the Japanese Overseas Technical Co-operation Agency (1973), and a Master Plan formulated for further water resource development in the basin.

A comprehensive water resource availability study was carried out by Mott MacDonald (1981). The study dealt with water use in the K. Surabaya and with water resource availability for expansion of city's water treatment facilities. The study concluded that water resource development in the basin had reached its limits and recommended a revision of the operation rules of Karangkates reservoir.

Japan International Co-operation Agency (1985) completed a water resource development and management plan for the basin. The study reviewed the 1973 Master Plan and analysed the opportunities for future development. An action plan was prepared covering all aspects of integrated development in the basin.

Mott MacDonald (1988) carried out a major water resources Master Plan study for water supply for East Java Province. As part of the project, a simulation model was developed for the Brantas Basin. Wardlaw (1993) describes the development of the model and the diverse ways in which the past developments have influenced the hydrological records in the basin.

5.2.2 The Climate Conditions

The Brantas Basin lies in the Southern Hemisphere and has a tropical climate. The mean annual rainfall over the basin is about 2000 mm varying from 1500-1800 mm in the delta area to over 3000 mm on the south western slopes of G. Kelud. The climate is monsoonal and the wet season is associated with the northwest monsoon which generally extends from November through April. During this period, 80 percent of the annual rainfall can be expected. This seasonal nature of rainfall leads to marked seasonality of streamflow in the K. Brantas. The mean daily temperature is generally around 23.8 $^\circ$ C in Malang and around 27.5 $^\circ$ C in the Brantas Delta. Temperature varies little throughout the year. The relative humidity is high at all times at about 60-80%.

5.2.3 Karangkates Reservoir

Karangkates reservoir is the most important reservoir in the Brantas Basin. It has a total catchment area of 2200 km^2 and regulates flows during the dry season. The effective storage capacity is 267 Mm^3 and although this is less than 4 percent of mean annual basin runoff, its regulating effect on dry season flows is significant. The releases from Karangkates reservoir are used for generating hydropower, for irrigation and for supplying drinking water to the City of Surabaya. For power production, Karangkates reservoir is very important. It has a total installed capacity of 105 mega watts (MW) and in 1987 accounted for 17.5 percent of the total installed capacity in East Java.

The characteristics of Karangkates reservoir are summarised in Table 5.1 and the elevation versus area and storage data is presented in Table 5.2.

Table 5.1 Characteristics of Karangkates dam and reservoir

Dam Characteristics	
Type	Rock Fill
Height	100.0 m
Crest Elevation	279.0 m
Crest Length	810.0 m
Reservoir Characteristics	
Catchment Area	2050 km^2
Normal HWL	272.5 m
Normal LWL	246.0 m
Design HWL	275.5 m
Abnormal HWL	277.5 m
Surface Area (HWL)	15.0 Km^2
Gross Storage Capacity	350.11 Mm^3
Effective Storage Capacity	267.0 Mm^3
Power Characteristics	
Type of Turbine	Vertical Francis
Installed Capacity	3 x 35 MW
Maximum Turbine Discharge	51.4 m^3/s
Minimum Turbine Discharge	25.0 m^3/s

Table 5.2 Elevation versus storage and area data for Karangkates reservoir

Elevation (masl)	Storage (Mm^3)	Area (Km^2)
215	0.922	0.37
220	4.18	1.01
225	10.40	1.62
230	20.02	2.38
235	33.76	3.30
240	52.61	4.54
245	77.34	5.78
246	83.40	5.98

250	107.67	6.80
255	143.77	8.15
260	186.94	9.76
262.5	212.56	11.13
265	241.10	12.65
267.5	273.51	14.12
270	309.73	15.70
272.5	350.11	17.46

5.2.4 Wonorejo Reservoir

Wonorejo Dam is being constructed on the K. Gondang to the west of Tulungagung. The location of Dam is shown in Figure 5.1. An inter-basin transfer will be provided from the K. Song by Segawe Weir which will have the effect of increasing the catchment area of the reservoir to 126 km^2. Wonorejo Dam will be a full storage reservoir producing only offpeak power, as there is no re-regulation downstream of Wonorejo. The principal characteristics are summarised in Table 5.3 and the area versus storage and elevation data is given in Table 5.4.

Table 5.3 Characteristics of Wonorejo dam and reservoir

Dam Characteristics	
Type	Rock Fill
Height	97.0 m
Crest Elevation	187.0 m
Crest Length	500.0 m
Reservoir characteristics	
Catchment Area	126.3 km^2
Normal HWL	183.0 m
Normal LWL	141.0 m
Surface Area (HWL)	3.4 Km^2
Gross Storage Capacity	122.0 Mm^3
Effective Storage Capacity	106.0 Mm^3

Power Characteristics	
Type of Turbine	Vertical Kaplan
Installed Capacity	1 x 6.2 MW
Maximum Turbine Discharge	10 m^3/s
Minimum Turbine Discharge	4.0 m^3/s

Table 5.4 Elevation versus storage and area data for Wonorejo

Elevation (masl)	Storage (Mm^3)	Area (Km^2)
130	6	0.6
135	9	0.9
140	14	1.2
145	21	1.5
150	30	1.8
155	40	2.1
160	50	2.4
165	63	2.7
170	78	3.0
175	93	3.3
180	110	3.6
183	122	3.8

5.2.5 Beng Reservoir

Beng reservoir is planned on the K. Beng about 5 Km from its confluence with the K. Brantas (Figure 5.1). The available storage capacity is quite high at the site. In order to utilise its full potential, the reservoir would be operated as an offstream storage for the K. Brantas. During periods of high flows, the water from the main stream will be pumped back to Beng and during periods of shortage water could be withdrawn from Beng to satisfy the irrigation demands.

Only offpeak power could be generated from Beng as the releases downstream of the reservoir cannot be regulated. A combined efficiency of 85% and a minimum turbine discharge of 2 m^3/s have been assumed for the power plant. The characteristics of Beng dam and reservoir have been described in Table 5.5.

Table 5.5 Characteristics of Beng dam and reservoir

Dam Characteristics	
Type	Earth Fill
Height	48 m
Crest Length	170.0 m
Reservoir Characteristics	
Catchment Area	134 km^2
Normal HWL	73.0 m
Normal LWL	52.0 m
Surface Area (HWL)	13.0 Km^2
Gross Storage Capacity	160.0 Mm^3
Effective Storage Capacity	147.0 Mm^3
Power Characteristics	
Type of Turbine	-
Installed Capacity	1 x 12 MW
Maximum Turbine Discharge	48.0 m^3/s
Maximum effective Head	31.0 m
Tail Water Level	35.0 m
Pumping Characteristics	
a) River to Canal	
Maximum Pumping Head	6.5 m
Pump Capacity	0.83 MW
Maximum Discharge	9.67 m^3/s
Pipe Length	37.5 m
Pipe Diameter	2.5 m
b) Canal to Reservoir	
Maximum Pumping Head	36.0 m
Pump Capacity	5.1 MW
Maximum Discharge	9.67 m^3/s
Pipe Length	2000 m
Pipe Diameter	23 m

The area versus elevation and storage data for the reservoir is shown in Table 5.6.

Table 5.6 Elevation versus storage and area data for Beng

Elevation (masl)	Storage (Mm^3)	Area (Km^2)
45	2	1.0
50	9	2.0
60	45	5.6
65	76	8.0
70	124	11.0
73	160	12.0

The upper and lower bounds on storages and releases from the reservoir are presented in Table 5.7 below.

Table 5.7 Lower and upper bounds for storages and releases

Reservoir	Allowable Storage (Mm^3)		Allowable Release (m^3/s)	
	Minimum	Maximum	Minimum	Maximum
Karangkates	83.4	350.11	25.0	155.0
Wonorejo	15.4	122.0	1.0	10.0
Beng	16.2	160.0	5.0	45.0

5.3 Water Use in the Basin

This section describes the main uses of water in the basin.

5.3.1 Irrigation

The primary use of water in the basin is for irrigated agriculture. The annual diversion requirement for schemes fed by the Brantas main stream alone is 1600 Mm^3. The irrigation water requirements from the basin were evaluated during Surabaya Water Use Study carried out by Mott MacDonald (1981). The study was based partly on the historic usage and partly on the cropping pattern in 1980.

5.3.2 Industrial Water Use

The Surabaya Water Use Study conducted by Mott MacDonald (1981) had put the industrial water use by direct abstraction from the K. Surabaya at 1.0 m^3/s after allowing for unlicensed abstractions and inaccuracies in the estimates of actual consumptive use.

5.3.3 Potable Water Demand

Surabaya is the commercial and industrial centre of East Java. The city has been expanding rapidly and so is the demand for potable water supplies. The demand projection for the City of Surabaya is shown in Figure 5.2. By the year 2000, the potable water demand was expected to rise to 9 m^3/s (Mott MacDonald 1988).

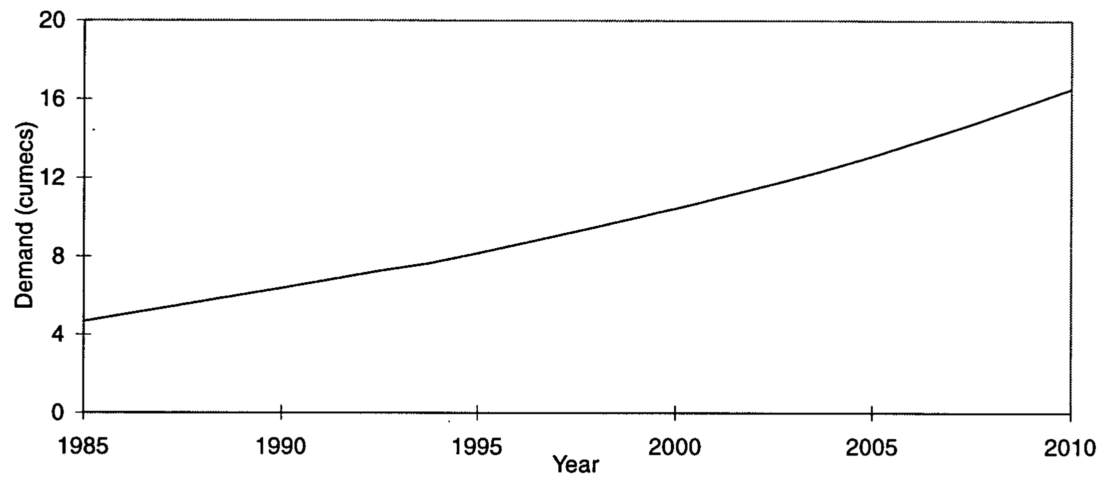


Figure 5.2 Potable water demand forecasts for City of Surabaya

5.3.4 City Water Way Flushing

Regular Flushing of canals in the urban and the semi urban parts of Surabaya is required to maintain an acceptable level of water quality and to prevent the deposition of organic matter on the bed of the canals. The demand for flushing irrigation canals in the City of Surabaya has been estimated to be of the order of 120 Mm^3 annually.

For the present research, it is the total demand from the river that is of importance. This includes the industrial demand, potable demand, flushing demand, evaporation, evapotranspiration and seepage losses. For the present time (1998/99), a potable demand of 8 m^3/s may be used. This gives a total demand of 16.5 m^3/s for non irrigation uses.

5.4 The Brantas Basin Simulation Model

General-purpose system simulation models are available (e.g., HEC-6, BTA-155 etc.) but they cannot be expected to cater to the needs of all types of river systems. A specific model for the basin was developed by Mott MacDonald (1988) as part of the East Java Provincial Water Resources Master Plan study for water supply. Wardlaw (1993) has described the development and application of the model for investigating alternative strategies for water resource development in the basin. The requirements of the model were that it should be able to evaluate the water resources in the basin under various scenarios of future development. Such scenarios included:

- changes in potable water demands;
- changes in irrigation water use;
- changes in water management techniques;
- changes in reservoir operation criteria;
- introduction of new storage schemes.

The model works by effectively routing a series of inflows through the network of natural river channels and reservoirs. It uses a network comprising nodes and reaches. At any node, flows may be diverted out of the system for irrigation or potable supplies subject to the availability of water. The influence of such diversions on downstream users is implicit in the modelling procedure. The model operates the reservoirs using predefined operating rules keeping track of reservoir levels, storages and surface areas for each 10 day time step. Based upon these values, the model can then work out the hydropower benefits and agriculture benefits in economic and financial terms for a specified reservoir operating strategy (Wardlaw 1993). There is a provision in the model to analyse the effect of introducing new storage schemes in the basin. Model runs can be carried out by including one or more potential reservoirs in the system and the performance of the system can be analysed.

5.4.1 Major Model Components

The simulation model consists of following components:

- reach inflows;
- storage reservoirs;

- run of river hydropower installations;
- pumping stations;
- irrigation diversions;
- potable water supply diversions;
- aqua culture diversions.

A model schematic representation of the river system without Wonorejo or Beng reservoirs is shown in Figure 5.3.

Reach inflows are natural local inflows to specific sections of the main river system and forms the driving data for the simulation model. These were evaluated using the existing streamflow and irrigation diversion records (Mott MacDonald 1988).

Storage reservoirs are those that have a regulating influence on the river and the physical characteristics of such reservoirs must be supplied to the model. For the Brantas Basin, Karangkates, Wonorejo and Beng are treated as storage reservoirs. Of these the effect of Karangkates on the river regulation is much more significant than the other two. The model performs a water balance on storage reservoirs during each time step of the simulation.

Run of river installations include Wlingi, Lodoyo Afterbay, Wangi and Sengguruh. Due to small effective storage at these installations, only single day regulation of river flows is possible. The only output from run of river hydropower nodes is peak and offpeak energy generated in each time step of the simulation.

Pumping schemes like the Ngrowo Push Back Scheme and Beng dam involve large pumping stations. At these nodes the model is required to compute the pump discharge and the energy required for pumping.

Irrigation schemes are represented as variable demand nodes in the model. At such nodes, the model attempts to satisfy the irrigation water demand in the current time step subject to water availability. Based on the actual water supplies, the model computes the agricultural production and the corresponding economic returns.

Potable water supply demands are also an input to the model. In periods of water shortage, priority is given to potable water supply over all other demands. Demands for aquaculture are treated by the model as a variable demand in a manner similar to the irrigation demands.

The model evaluates the fisheries production as a function of actual water supplied to the fisheries.

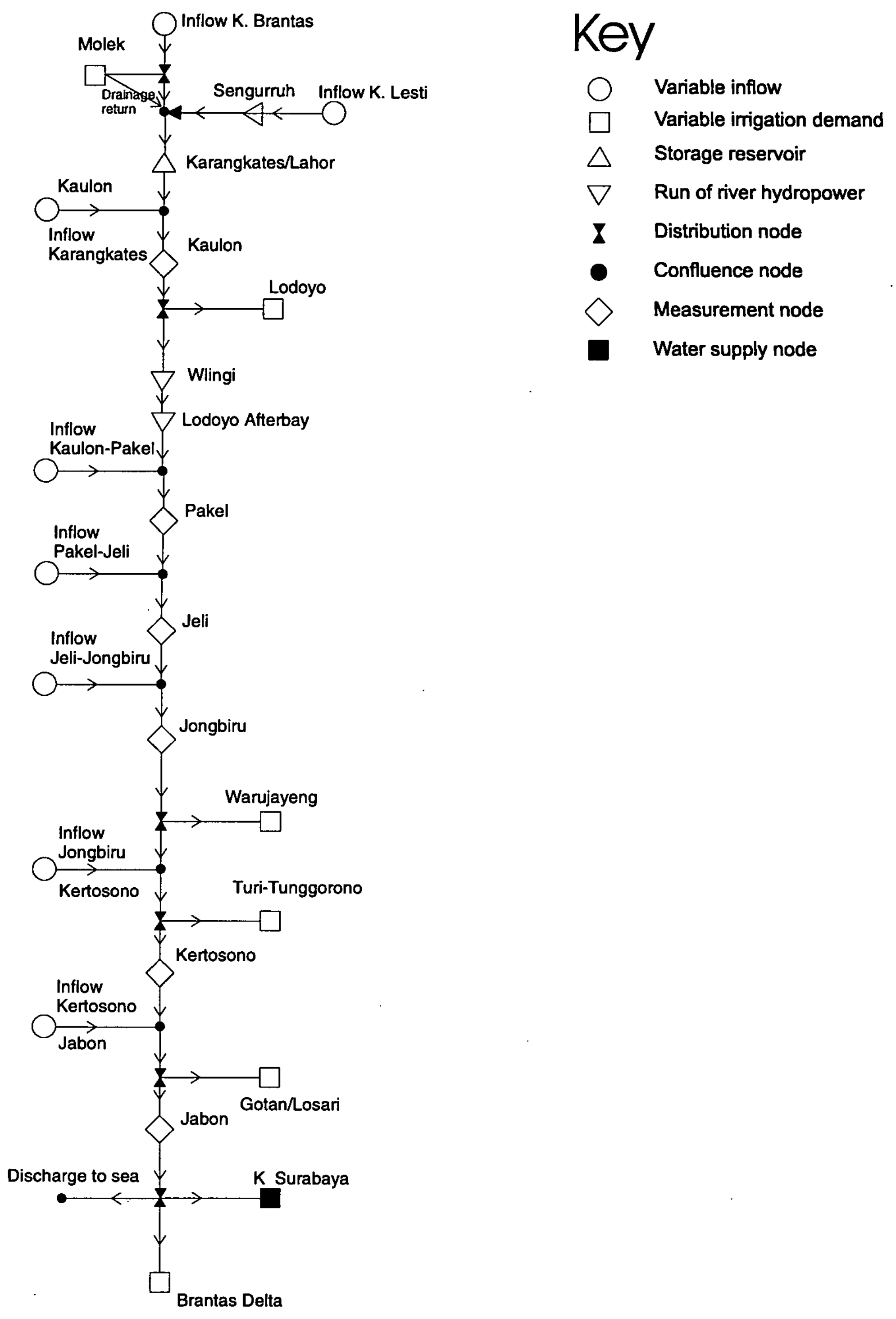


Figure 5.3 Model network, existing development situation

5.4.2 Types of Nodes Included in the Model

In all, 11 different node types have been included in the model

1. Variable inflow node
2. Confluence node
3. Multiple diversion node
4. Fixed head pumping node
5. Variable head pumping node
6. Storage reservoir node
7. Run-of-river hydropower node
8. Irrigation or fisheries demand node
9. Potable water supply demand node
10. Total demand node
11. Decision node

The inflows must be described for each variable inflow node. The model does not carry out any computation at such nodes. At a confluence node, the model combines all the inflows to it from the specified upstream nodes and then passes a single outflow downstream. The model allows for upto five upstream nodes to contribute to a confluence node.

At a multiple diversion node, water is diverted to irrigation demand nodes and to potable water demand nodes. For irrigation and potable water demand nodes the model attempts to satisfy the specified demand of the node in the current time step, subject to water availability. The residual is passed on downstream after satisfying the demands.

At the fixed pumping head node, the model computes the energy required for pumping the water to schemes like Ngrowo Push Back and Beng Pumped Storage Scheme. At a variable pumping node, the delivery is to a reservoir in which the water level will vary from one time step to another. The model receives the current reservoir level from the delivery node, and computes the pumping head.

At a storage reservoir node, full water balance is carried out on the reservoir taking into account the physical characteristics of the reservoir and the operational criteria. The outflow from the reservoir is passed on to the next node downstream. Karangkates, Wonorejo and Beng are treated by the model as storage reservoirs.

For run-of-river hydropower nodes, a water balance for each time step is carried out using average climatic data at the site. All the inflows into the nodes are summed up and outflow is computed by taking off the reservoir losses.

At potable water supply nodes, the model does not take any computational action and such nodes act only as demand from the main system that must be met. At a diversion node, the model allows diversions to be made from the main network to satisfy a combination of water demands. These may include water supply, irrigation, river flushing etc. Priorities can be specified for each of these uses and in periods of shortage, allocations made accordingly.

A decision node is a node at which the model must make decisions based upon the given criteria, whether or not certain diversions are to be made and whether pumping from a particular node is to be invoked. Decision nodes are linked to the dynamic mode of operation in the model with which reservoir releases and pump station operation are made in consideration of the forecast of expected catchment response for the next time step.

5.5 Simulation Model Runs

This section presents some simulation model runs. The Brantas Basin simulation model was set up to run in a continuous simulation mode for the entire period of available streamflow data (1950 - 1986). The existing situation in the basin (in 1987) was represented by the network shown in Figure 5.3. It was assumed that for the existing situation in the basin, the operation of Karangkates reservoir is according to a rule curve derived in 1978 (Nippon Koei 1978). This rule curve is known as 1978 rule curve, and is shown in Figure 5.4. The rule curve specifies the target levels to be maintained at the end of each 10-day time step. To evaluate the existing situation in the basin, a base run employing the 1978 rule curve was carried out using the calculated theoretical irrigation requirements for present day cropping practices. The results are presented in Figure 5.5 to Figure 5.8.

Figure 5.5 shows the simulated irrigation deficits at various exceedence probabilities with the current operating rule. The model evaluates deficits in each 10 day time-step by satisfying all demands from upstream to downstream and then comparing the demands of the Brantas Delta with the available river flow. From Figure 5.5 it is clear that during dry periods there are large deficits even with 80% exceedence probabilities.

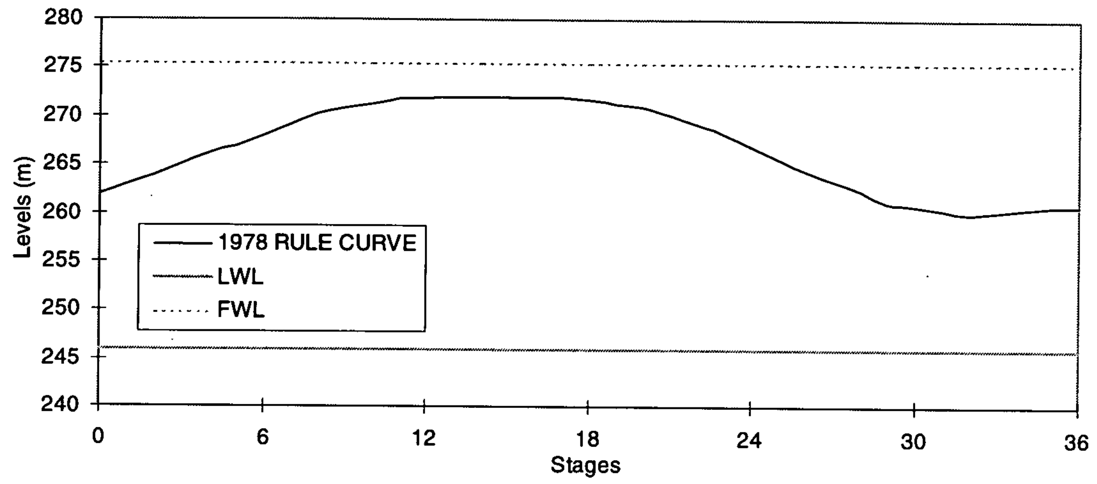


Figure 5.4 The 1978 rule curve

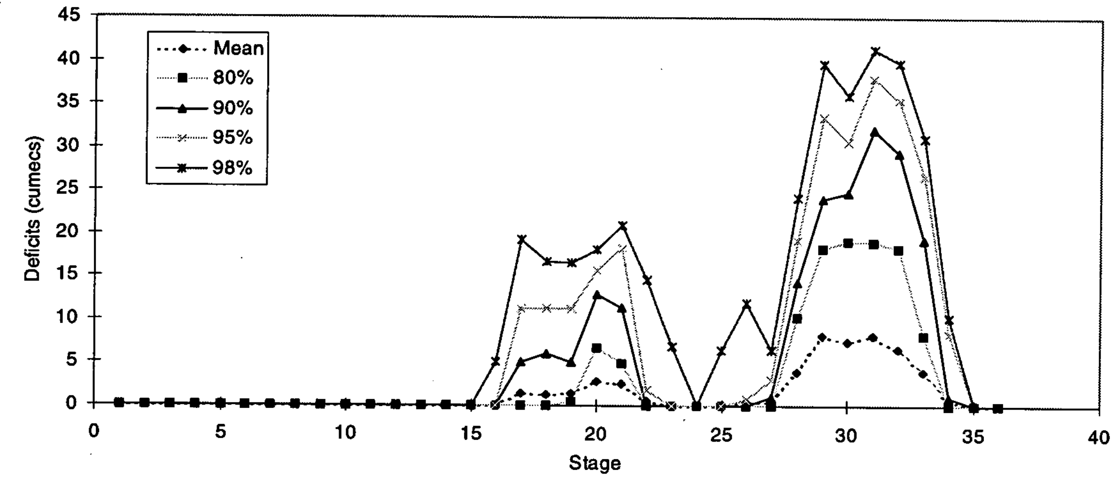


Figure 5.5 Simulated irrigation deficits at different exceedence probabilities with 1978 rule curve

Figure 5.6 shows the simulated water levels in Karangkates reservoir with the 1978 rule curve, from which it appears that reservoir storage is not fully utilised. It was seen that with the 1978 rule curve there is a failure to realise full potential of agricultural production, and the mean annual damages are of the order of 5000 million Rupiah. The 1978 rule curve was intended primarily to stabilise power production and in that respect it has served its purpose well. Figure 5.7 and Figure 5.8 show offpeak and peak power production at Karangkates with the 1978 rule curve.

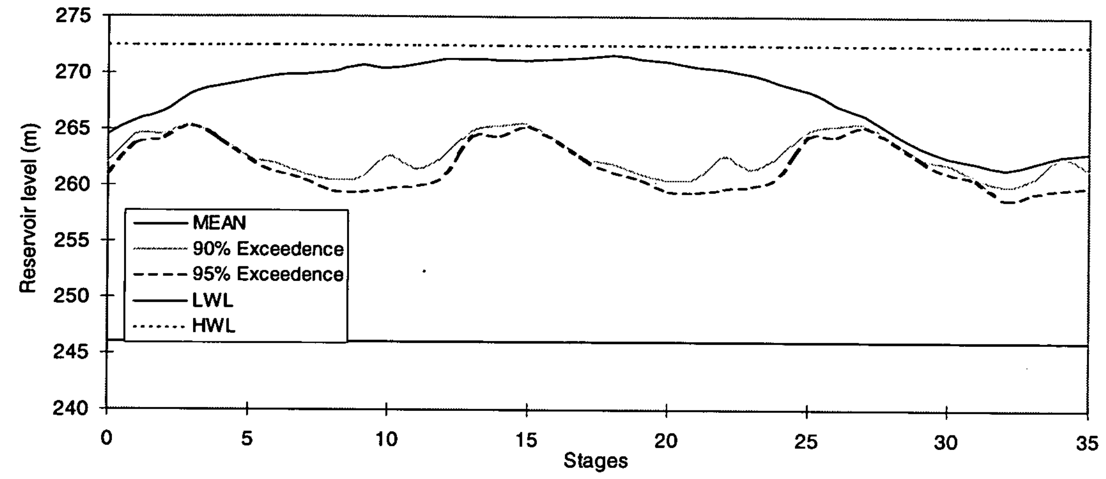


Figure 5.6 Karangkates water levels , base run

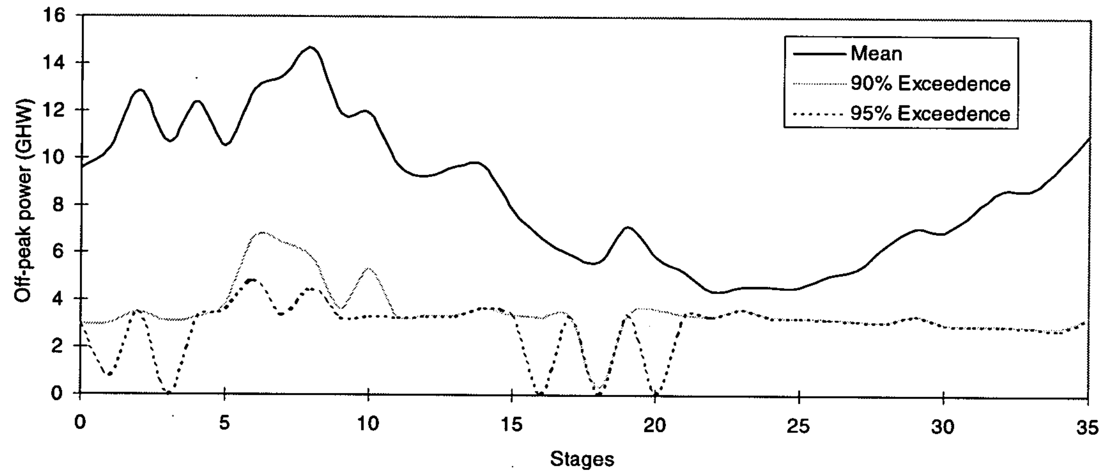


Figure 5.7 Karangkates offpeak power, base run

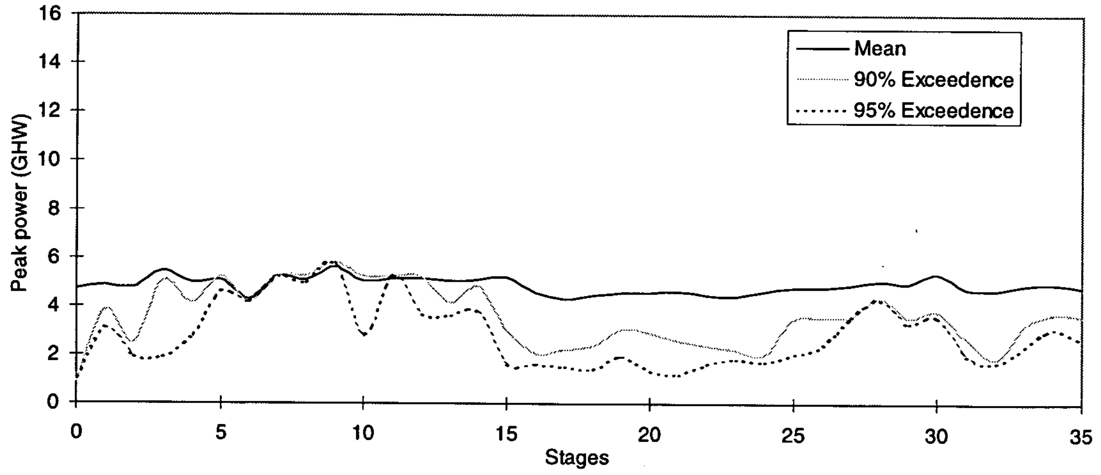


Figure 5.8 Karangkates peak power, base run

5.6 Conclusions

In this chapter, the characteristics of the Brantas Basin and the storage reservoirs in the basin were described. A brief survey of the past water resource development studies in the basin was also presented. The development of the simulation model has also been described in this chapter. Model runs carried out in the evaluation of the existing situation in the basin were also presented. In the next chapter, a number of development options in the basin given by Mott MacDonald (1988) are considered. The development of optimal operating rules for two such future water resource development scenarios in the basin is also described. The simulation model described here is used in the next chapter for assessing long term economic impacts of alternative scenarios of water resource development, and in this sense is used conjuctively with the optimisation models.

6. APPLICATION OF GENETIC ALGORITHMS TO THE BRANTAS BASIN

6.1 Introduction

This chapter describes application of the GA and DDDP approaches to the operation of the Brantas Basin reservoir system in Indonesia. The main objective in applying the GA to the Brantas Basin was to investigate the practicality of the approach in the optimisation of control curves for a real reservoir system. A series of sensitivity analyses have also been carried out to determine appropriate parameter settings for the GA. The characteristics of basin and of the storage reservoirs, along with the details of the simulation model for the basin have been described in the previous chapter.

The pressure on the available water resources in the basin has been increasing at a rapid rate as a result of large scale industrial and commercial development. This has led to an increased emphasis on the development of optimal operating strategies for the basin. Various development schemes were proposed by Mott MacDonald (1988), and these are shown in Figure 6.1. In this case study, operating rules have been derived for the existing development situation in the basin, and also for two future water resource development scenarios using the GA approach. Comparison of results achieved by using the GA approach has been made with those achieved by DDDP. The reason for choosing DDDP to compare the solutions obtained by GA is that the solutions found by DDDP are near optimal for a given number of states and release decisions. Another reason for choosing DDDP is the generality of the objective functions that can be treated with the approach. The derived operating rules are then fed into the simulation model of the basin to evaluate the long term economic impacts.

The GA approach has been applied to five cases. The first is the application of the approach to the situation with only Karangkates reservoir operating, and with optimisation carried out to maximise hydropower returns. In the second case, the optimisation is carried out for hydropower and irrigation returns, again with Karangkates reservoir only. The third case considers the optimisation with respect to irrigation production only. These three cases are

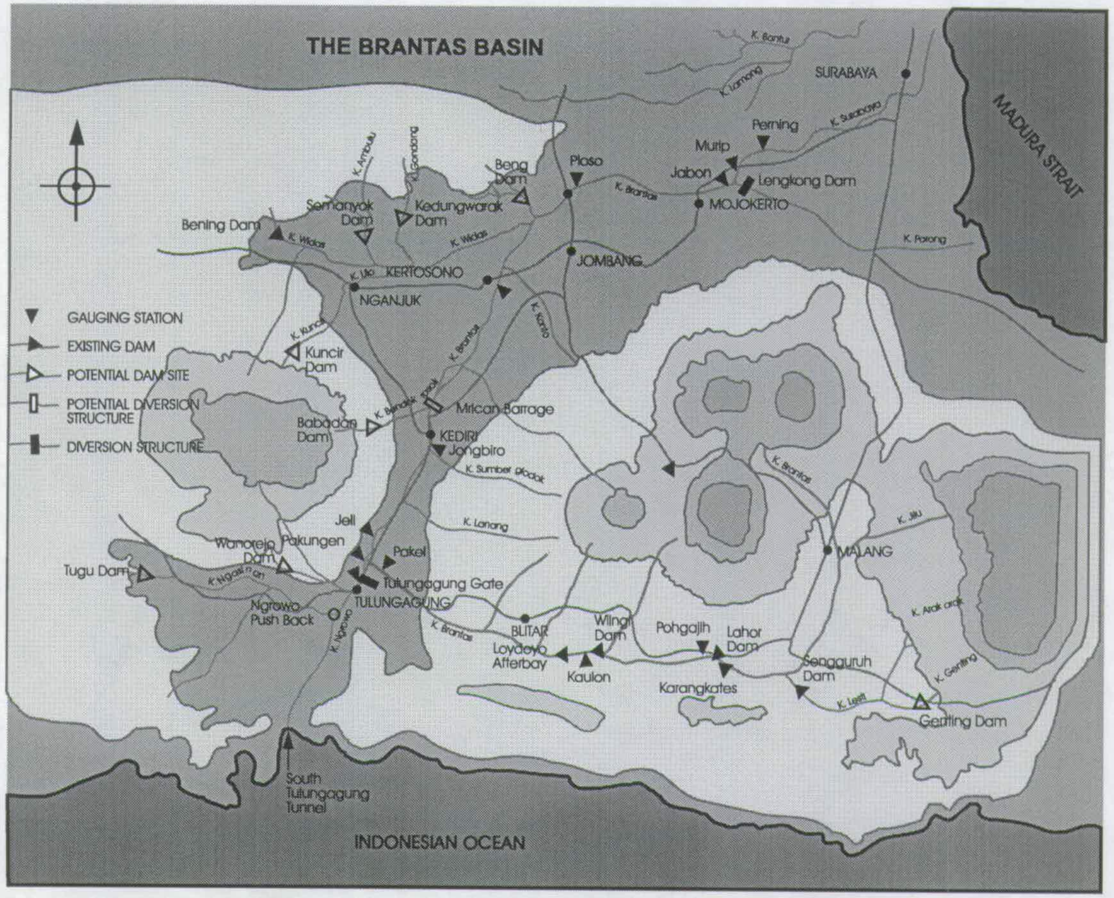


Figure 6.1 Development schemes in the Brantas Basin

described in section 6.3. The application of the GA approach to a future water resource development scenario in the basin that includes Wonorejo reservoir in the system is considered in section 6.4. Finally, operation of the reservoir system for another future scenario that includes both Wonorejo and Beng reservoirs in the system is described in section 6.5. For cases 4 and 5, both hydropower and irrigation benefits are considered. The sensitivity of GA performance to mutation and crossover probability is discussed in sections 6.6 and 6.7 respectively. The results of economic post processing are evaluated in section 6.8. Conclusions are presented in section 6.9.

6.2 System Dynamics

The dynamics of a multi reservoir system can be described by the following equation

$$S_i(t+1) = S_i(t) + I_i(t) + MR_i(t) - E_i(t) \tag{6.1}$$

where $\mathbf{S}_i(t)$ = vector of reservoir storages at time *t* in reservoirs *i=1, n*; $\mathbf{I}_i(t)$ = vector of reservoir inflows in time period *t* to reservoirs *i=1, n*; $\mathbf{R}_i(t)$ = vector of reservoir releases in time period *t* from reservoirs *i = 1, n*; $\mathbf{E}_i(t)$ = vector of reservoir evaporation in time period *t* from reservoirs *i= 1, n;* and **M** = $n \times n$ matrix of indices of reservoir connections. The matrix **M** depends upon the configuration of the system, and consists of -1, 0, and 1.

The transformation from stage to stage is governed by (6.1). In addition to this, the releases and storages are limited by physical considerations. A minimum turbine release is required to avoid cavitation, and there are restrictions on the maximum allowable release in any time period of the optimization horizon. Similarly, the storage in the reservoir should not fall below a specified minimum level nor should it exceed maximum allowable storage in any time step. The system is, therefore, subject to constraints expressed as follows:

$$S_{i,\min} \le S_i(t) \le S_{i,\max} \tag{6.2}$$

$$R_{i,\min} \le R_i(t) \le R_{i,\max} \tag{6.3}$$

where $S_{i,\ min}$ and $S_{i,\ max}$ are the allowable minimum and maximum storages in reservoir i = 1 ,n, and $R_{i,\ min}$ and $R_{i,\ max}$ are the lower and upper bounds on the releases from reservoir $i = 1, n$.

6.3 Model Application to the Existing Development Situation

This section discusses the derivation of control curves for Karangkates reservoir using the GA and the DDDP approach. Three cases are considered; (i) optimising for hydropower only, (ii) optimising for hydropower and irrigation, and (iii) optimising for irrigation benefits only. The operation of Karangkates reservoir was originally carried out according to a rule curve derived by the Japanese Overseas Technical Co-operation Agency (1973). It was based on the assumption that during the wet season the hydropower production should be maximised, while during the dry season the irrigation demand should have a priority over the hydropower production. This operation was revised by Nippon Koei (1978) to an objective of maximising the hydropower production all the year round. This operating rule was developed as a rule curve giving target levels at the end of each 10-day operating period, and is known as 1978 rule curve.

The historic inflows and the other data used in this study come from Mott MacDonald (1988). The flow records for the K. Brantas are available from 1950 onwards. The Karangkates reservoir was constructed in 1972, and since then it has significantly influenced the natural flow regime of the K. Brantas. Since historic reservoir operation may not be typical of likely future operation, streamflow naturalisation has been carried out by Mott MacDonald (1988) to remove the influence of Karangkates reservoir from the historic flow records. This results in a data base on which various possible future development scenarios could be tested.

The DDDP and GA models are applied to a single year of data for a critical period with a view to determining control curves for the reservoirs. A critical inflow series with 90% annual exceedence probability was derived for Karangkates reservoir. Figure 6.2 shows the smoothed critical dependent, and independent inflow series for Karangkates. A log-normal distribution has been fitted to the series of 10-day inflows to Karangkates. The log-normal distribution fits the data well for most time steps but there are few time steps where the distribution is not particularly good. The probability distribution of inflows in each of the 36 durations is shown in appendix A of this thesis.

The tributary inflows between Karangkates and the Brantas Delta were combined to obtain a local inflow series giving flows upstream of Jabon (Figure 6.3). Critical period analysis was then carried on this series to obtain a local inflow series with 90% annual exceedence probability. A net local inflow series at Jabon was obtained by subtracting from this series the irrigation demands between the reservoir and the delta. The tributary inflows and irrigation diversions are shown in Figure 6.3. The irrigation demands and the demands at the Brantas Delta are based on the calculated irrigation diversion requirements as determined by Mott MacDonald (1988). The optimisation models use a potable demand of 8 m^3/s. The supply to the delta is calculated by the model in the following manner.

$$Q_{delta} = Q_{karangkates} + Q_{net\_local\_\inf} - Q_{total\_demand} \tag{6.4}$$

where Q_{delta} is the water supplied to delta, $Q_{karangkates}$ is the release from the reservoir, $Q_{net\_local\_inf}$ is the net local inflow upstream of Jabon, and Q_{total_demand} is the total abstractions from K. Surabaya including the potable water demand of the City of Surabaya.

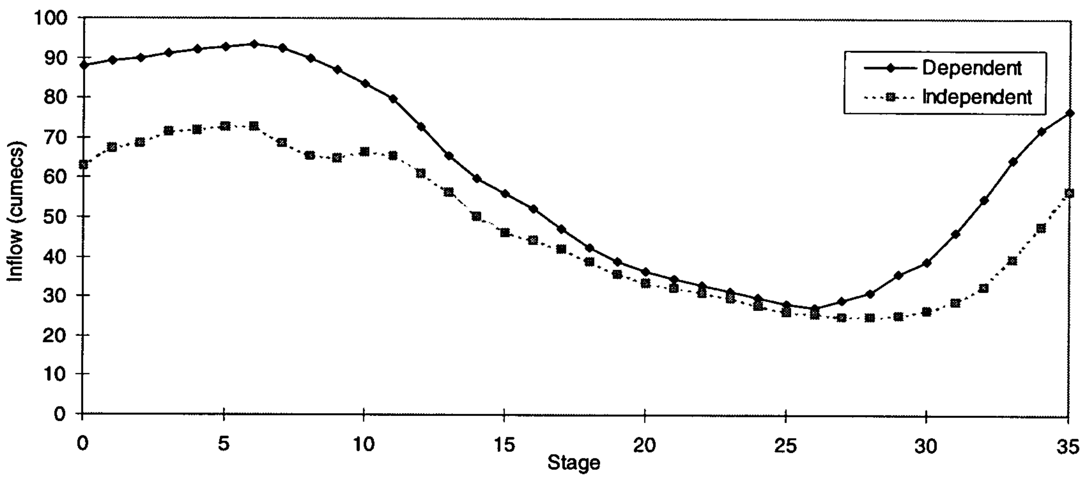


Figure 6.2 Critical dependent, and independent inflow series for Karangkates

The control curves for Karangkates were developed based on the derived critical inflow sequences using the GA and the DDDP approaches. These curves were then applied in a simulation model of the Brantas Basin to obtain estimates of long term economic benefits from operation of the reservoir system. The network used by the models is shown in Figure 6.3. With only Karangkates reservoir in operation, there are huge deficits in irrigation supply even with a potable demand of 4 m^3/s, although the current potable demand is probably of the order of 8 m^3/s. The objective of modelling is to assess the impact of new development schemes, such as Wonorejo and Beng, on the deficits in the irrigation water supply while taking into account the current potable demand for the City of Surabaya.

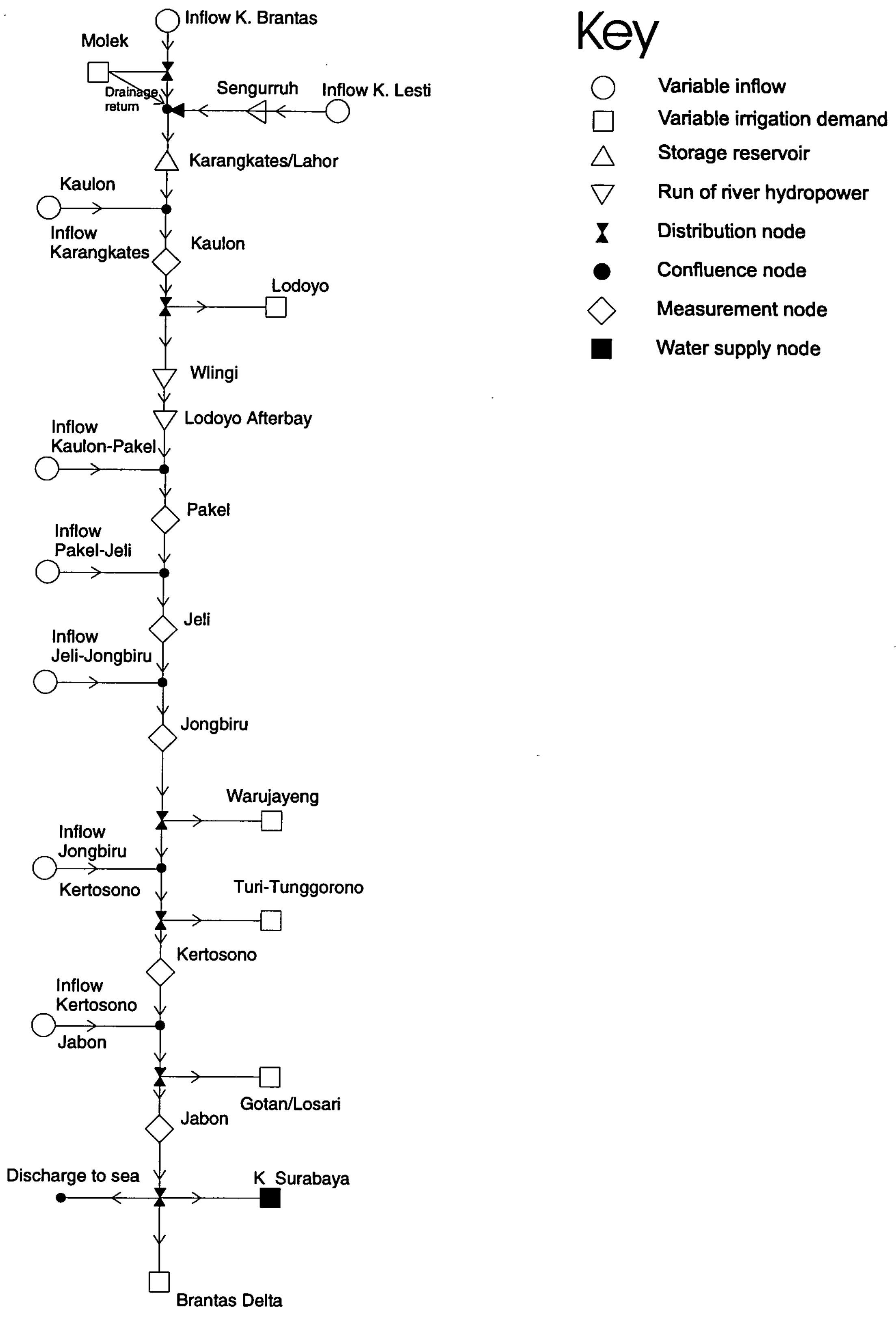


Figure 6.3 Existing development situation in the Brantas Basin

6.3.1 Existing Development Situation 1 (EDS1): Optimising for Hydropower only

In this case, the optimisation has been carried out for 36 10-day time steps with the sole objective of maximising the hydropower production while allowing deficits to occur in irrigation supplies. In view of the importance of Karangkates reservoir, it was important to be able to accurately model the power produced for peak and off-peak generation. A series of polynomial equations was derived by Nippon Koei (1978) to describe the waterway head loss - discharge, and tail water level - discharge relationships for the power station. Using these equations, the power generated can be calculated for the releases made from the reservoir. The general form of the equations used for calculation of waterway head loss, tail water elevation and power are as follows.

Water way head loss

$$H_{whl} = A_{0w} + A_{1w} \times Q + A_{2w} \times Q^2 + A_{3w} \times Q^3 \qquad (6.5)$$

Tail water level

$$H_{twl} = A_{ot} + A_{1t} \times Q + A_{2t} \times Q^2 + A_{3t} \times Q^3 \qquad (6.6)$$

Power

$$P = A_{0p} + A_{1p} \times H_{eff} + A_{2p} \times Q + A_{3p} \times Q \times H_{eff} \qquad (6.7)$$

The effective head required for computation of power produced can be calculated as follows.

$$H_{eff} = H_{avg} - H_{twl} - H_{whl} \qquad (6.8)$$

where $A_{0w}, A_{1w}, A_{2w}, A_{3w}$ are the coefficients of waterway head loss calculation, $A_{0t}, A_{1t}, A_{2t}, A_{3t}$ are the coefficients of tail water level calculation, $A_{0p}, A_{1p}, A_{2p}, A_{3p}$ are the coefficients of power calculation, H_{whl} is water way head loss, H_{twl} is tail water elevation, H_{eff} is effective head, H_{avg} is average water level in metres in the reservoir during the current time step, Q is discharge per turbine unit in m^3/s, and P is power produced in MW. The coefficients of (6.5), (6.6), and (6.7) are given in Table 6.1.

Table 6.1 Coefficients for computation of hydropower

COEFFICIENTS	KARANGKATES	WONOREJO	BENG
A_{0w}	0.0	0.0	0.0
A_{1w}	0.02875	0.0	0.0
A_{2w}	0.12	0.0	0.0
A_{3w}	0.00125	0.0	0.0
A_{0t}	179.0	110.5	35.0
A_{1t}	0.018	0.0	0.0
A_{2t}	0.0	0.0	0.0
A3t	0.0	0.0	0.0
A_{0p}	1.64465	0.0	0.0
A_{1p}	-0.0847	0.0	0.0
A_{2p}	-0.015149	0.0	0.0
A_{3p}	0.012193	0.00834	0.00834

For Karangkates reservoir, a minimum reservoir level of 246 metres is required for hydropower generation. If the water level drops below this value then no power can be produced because of the possibility of cavitation. The financial value of peak power is twice that of offpeak power. If there is not enough water to release for full peak and off-peak power generation, then peak power is generated in preference to off-peak power. Peak power is generated for 5 hours per day, the remainder being offpeak. The objective is to maximise the hydropower returns from the reservoir subject to the given constraints. An additional constraint is that the state of the system at the end of the control horizon must be same as that at the beginning of the control horizon. The objective function may be expressed as

$$Maximise\ Z = \sum_{i=1}^{n} \sum_{t=1}^{N} E_{it}(\overline{S}_{it}, R_{it}) \tag{6.9}$$

where n is the number of reservoirs included in the model, N is the operating horizon, $E_{it}(\overline{S_{it}}, R_{it})$ is the economic return from the power produced from reservoir $i = 1, n$, during period $t = 1, N$; $\bar{S}_{it}$ is the average storage in reservoir i during time step t, and R_{it} is the release made from reservoir i in time step t.

For the calculation of power produced, the effective head is required. The reservoir level is continually changing as both inflows occur and releases are made. Since the reservoir state is discretized and only one power calculation is made for each 10 day operating period, the power functions use the average reservoir level for each stage. The average of the state before and after a release is made is used to determine the average reservoir level in a particular time step. The values of H_{whl}, H_{twl}, and H_{eff} are computed from (6.5), (6.6), and (6.8) respectively. Finally, the power produced can be computed using (6.7).

6.3.2 Existing Development Situation 2 (EDS2): Optimising for Hydropower and Irrigation

For this case the objective function comprises the hydropower benefits and a penalty for failing to supply the required water for irrigation. The model attempts to minimise the deficits in irrigation supplies while maximising the hydropower production. The agricultural production was included into the GA and the DDDP models through an imposed penalty function. The penalty function is quadratic and is given by (6.11). The model does not include the potable water demand as a benefit in the objective function but Surabaya City potable water demand has to be met before any water is diverted for irrigation in the Brantas Delta. If the demand is not met, then agriculture would receive no water thus causing a large penalty on the return function. The potable water demand is therefore included in the objective function through the imposed constraints. The objective function for this case can be expressed as

$$Maximise\ Z = \sum_{i=1}^{n} \sum_{t=1}^{N} \left[E_{it}(\bar{S}_{it}, R_{it}) - P_{it} \right] \tag{6.10}$$

whereP_{1t} = Penalty for not meeting the irrigation demand for the delta assumed to be of the following form.

$$P_{1t} = k \times \left[d_{1t} - x_{1t} \right]^2 \tag{6.11}$$

where k is penalty factor, d_{1t} is irrigation demand at delta, and x_{1t} is irrigation supply to delta in time step t. The selection of penalty factor is also important as it determines the severity of penalty imposed. The objective function given by (6.10) is considered complex because hydropower production is a highly non-linear function of discharge and power head. Also, the objective function includes a penalty term.

6.3.3 Existing Development Situation 3 (EDS3): Optimising for Irrigation Benefits only

The objective in this case study is to minimise the deficits in irrigation water supply without taking into account the hydropower production. A penalty function similar to that used for EDS2 has been employed. The potable water demand is included in the model through the imposed constraints, and has a priority over the irrigation demands. The objective function may be expressed as follows.

$$Maximise\ Z = \sum_{i=1}^{n} \sum_{t=1}^{N} \left(C - P_{it} \right) \tag{6.12}$$

where C is a positive constant, and P_{1t} is the penalty for not meeting the irrigation demand for the delta assumed to be of the form given by (6.11).

6.3.4 The Solution Procedure

The solution to the problems described above were obtained using the GA and the DDDP approach. The essentials of the GA approach have been described in chapter 3. Real-valued representation with the tournament selection scheme, uniform crossover, modified uniform mutation and elitism is used. The chromosome length may be expressed as a product of the number of stages and the number of reservoirs in the system. For the four reservoir problem discussed in chapter 4, the chromosome length in terms of number of genes was 4 x 12 = 48 for 12 time steps, while that for a single reservoir in this system with 36 time steps it is 1 x 36 = 36. The decision variables are releases during each time period, $\mathbf{R}_1(t)$, t = 1, N. In a GA, the discretization of storage and release is not required. Each gene within the chromosome represents a release made from the reservoir and can take up any value between the upper and lower bounds of releases. The aim of the GA is to find a gene sequence that yields the best chromosome i.e. maximises the objective function values. In a GA, the discretization of storage and release is not required. A population size of 100 has been used and the model allowed to run for 1000 generations for all the three cases described above. A crossover probability of 0.7 and a mutation probability of 0.028, which corresponds to one mutation per chromosome, has been adopted.

The evaluation function used by GA corresponds to the objective functions given by (6.9), (6.10), and (6.12) for case EDS1, EDS2, and EDS3 respectively. The constraints on storages are handled by using a quadratic penalty function based upon the degree of constraint violation. The fitness of chromosomes is appropriately modified for any constraint violations. The GA initiates the search with a randomly generated set of solutions which may or may not be feasible. As the GA run progresses, the infeasible solutions are eliminated because the penalties assigned to them reduce their fitness, thus making their chances slim for selection in the next generation. At the end of the run, a large number of feasible and near optimal solutions are obtained. Figure 6.4 shows the plot of generation number versus best-of-generation maximum normalised fitness and average fitness of the population for the objective of maximising hydropower returns (EDS1).

In all other cases to follow, the results shown are for a crossover probability of 0.7 and for a mutation probability corresponding to one mutation per chromosome irrespective of the length of the chromosome. This has been done so that the consistency in the presentation of results could be maintained although in some cases slightly better objective function values could be obtained for a different combination of crossover and mutation probability. It can be observed from Figure 6.4 that the GA starts with poor initial solutions but picks up quickly. By the end of 200 generations, a number of good solutions have been found. Once, the GA locates good solutions, some fine-tuning may be required to improve those solutions. The GA was therefore allowed to run for 1000 generations although the improvement has been found to be small after 500 generations. The variation of best-of-generation maximum fitness and the average fitness of the generation for case EDS2 and EDS3 is shown in Figure 6.5 and Figure 6.6 respectively. It can be observed from these figures that the manner in which the fitness progresses with the generation for EDS2 is very similar to the earlier case (case EDS1). In EDS3, the average fitness of the population quickly approached the maximum fitness. This is because a large number of solutions close to the optimum were identified by the GA in the early part of the run, and there was little scope for improvement thereafter.

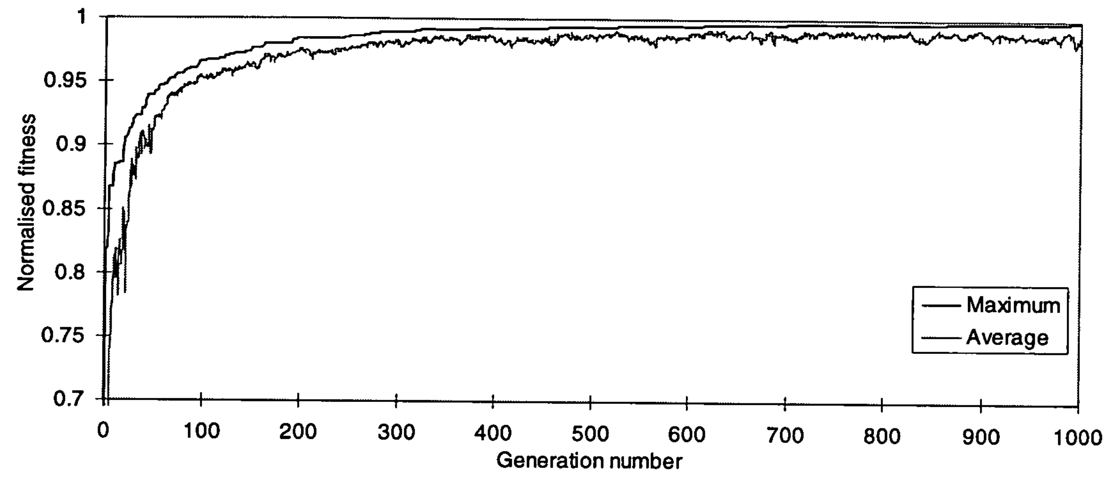


Figure 6.4 Generation versus maximum and average fitness, EDS1

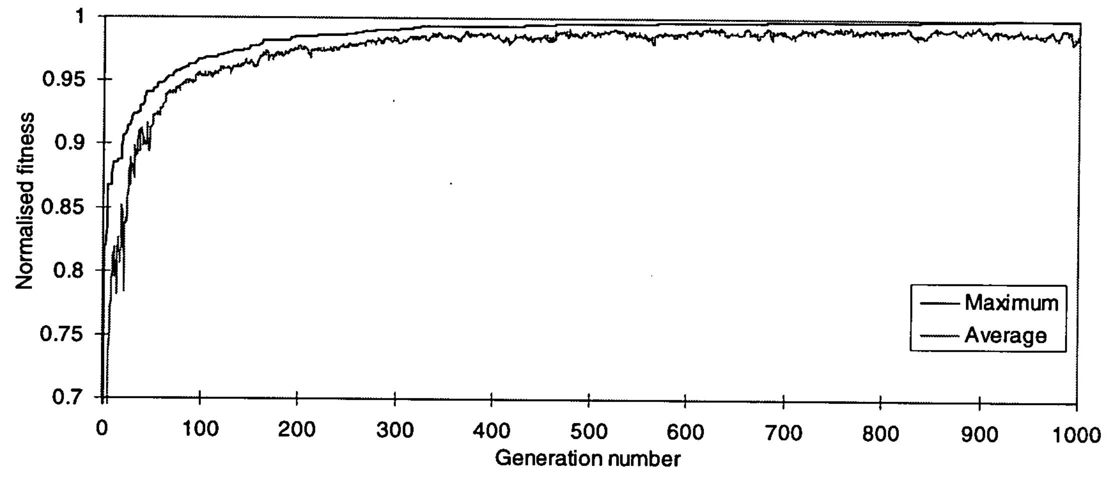


Figure 6.5 Generation versus maximum and average fitness, EDS2

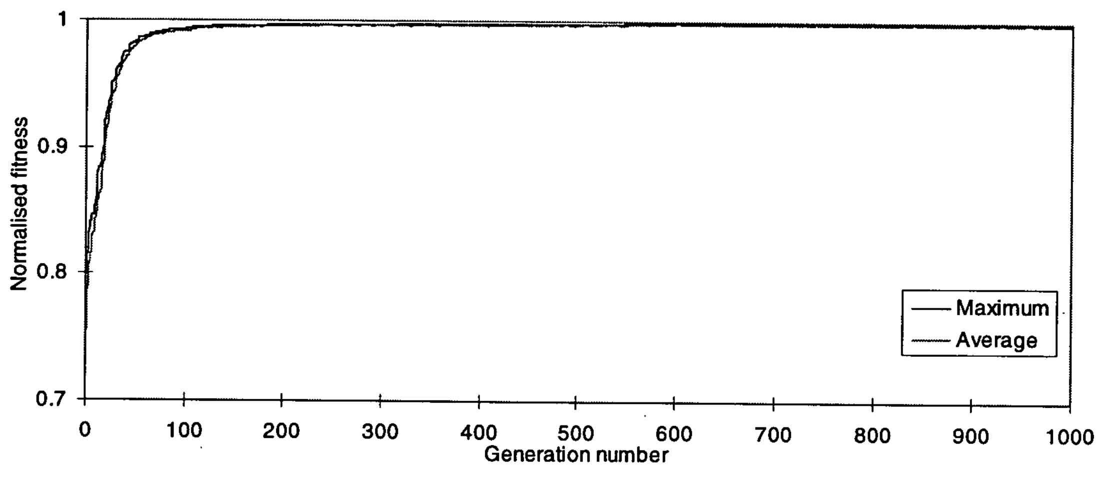


Figure 6.6 Generation versus maximum and average fitness, EDS3

All the three problems described earlier were also solved using DDDP so that an evaluation of the GA approach could be carried out. Instead of searching over the entire state-stage domain for the optimum, as is the case for DP, DDDP analyses only few states at each stage. DDDP starts with the selection of a trial state trajectory for the system and searches for improved trajectories in the neighbourhood of the trial trajectory. A corridor is defined around the current trajectory within which the optimisation is constrained. The traditional DP algorithm (Bellman 1957) is then used to find the optimal trajectory within the corridor. The recursive eqaution used was

$$F_n(s_n) = \max\ [V_n(s_n, d_n) + F_{n-1}(s_{n-1})] \tag{6.13}$$

where s_n is the state variable, d_n is the decision variable, $V_n\ (s_n, d_n)$ is the objective function value, $F_n(s_n)$ is the cumulative return at stage n with $F_0(s_0)$ known, and $s_{n+1} = g(s_n, d_n)$ is the stage to stage transformation function. An improved trajectory is obtained after each iteration and used as a trial trajectory for the next iteration. The procedure continues until no further improvement in the value of objective function is obtained. DDDP reduces the execution time significantly by analysing only a small subset of all feasible states at each stage.

Application of DDDP requires that the problem be divided into states and stages. For the problem under consideration, the stage was taken to be a 10 day period. The decision variables are the release made from the reservoir and the state variables are represented by the storage in the reservoir at the beginning of each stage. For the development of reservoir states, the storage volume of Karangkates reservoir is discretized into 100 states ranging from the minimum storage volume to maximum storage volume in increments of 3.535 Mm^3. The reservoir levels and areas were calculated for each of the 100 states using the available data. The original data for the storage volume to reservoir level does not provide enough information to generate levels for all storage volumes. In order to calculate the corresponding levels and areas for each state, linear interpolation between the existing data was used. The choice of discretization levels requires careful consideration in a DDDP algorithm. A coarser discretization may lead to inaccurate results and a finer discretization may be computationally intensive. In any case, the discretization need not be finer than the actual controls at the reservoirs. For reservoirs with huge storage capacity, a large number of discretizations may be required to represent the problem adequately. The GA approach treats the storage space as continuous, and is unaffected by the size of the reservoirs.

A comparison of the trajectories produced by the GA and by DDDP for EDS1 is presented in Figure 6.7. The trajectories obtained by the GA matches closely to that of DDDP except for a few points. In situations of multiple local optima, the optimum objective values obtained by different techniques may not differ much in magnitude but the storage or level trajectories can be significantly different. These rule curves were obtained for a 10-year critical drought sequence and give the target reservoir levels at the end of 10-day operating periods. Application of such a rule implies that as får as possible target reservoir levels should be maintained. If the inflows are such that available water results in levels higher than required by the rule curve the surplus water may also be released downstream. On the other hand, if the reservoir level falls below the target values, the release are curtailed in subsequent stages until the desired levels are achieved. It can be seen that the GA trajectories follow closely those produced by the DDDP.

When the optimisation was carried out for both power and irrigation, the target levels produced by the GA and the DDDP models are lower than for the case when the optimisation was done for power only. This is because the models in case EDS2 attempts to satisfy the irrigation demands of the delta in addition to maximising the hydropower returns, thus lowering the reservoir levels particularly during the drier periods (Figure 6.8). The trajectories obtained for the case when the optimisation was carried out for maximising agricultural production only are shown in Figure 6.9. These trajectories suggest that reservoir should be maintained at higher levels during wet periods while higher amounts of release should be implemented during drier periods. It can be observed from the trajectories that in EDS3, the reservoir does not draw down to levels lower than in EDS1 and EDS2 during drier periods. It can be observed from the trajectories presented in Figure 6.9 that during the first half of the operating horizon, the reservoir keeps filling up and the levels become considerably high. During the dry periods, the models release just enough water to satisfy the irrigation demands. Owing to already high levels in the reservoir, these releases during the dry periods do not result in levels lower than in EDS1 or EDS2.

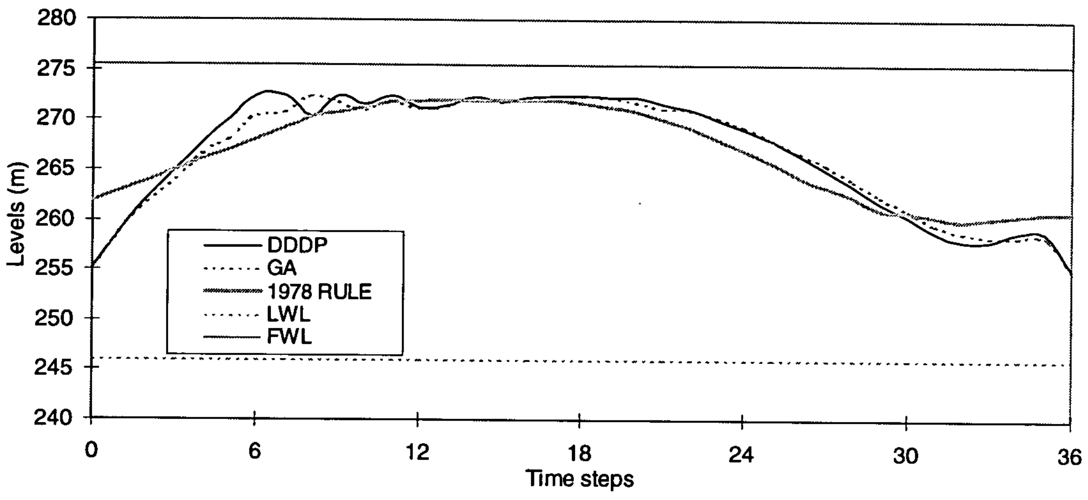


Figure 6.7 Level trajectories, optimising for hydropower only

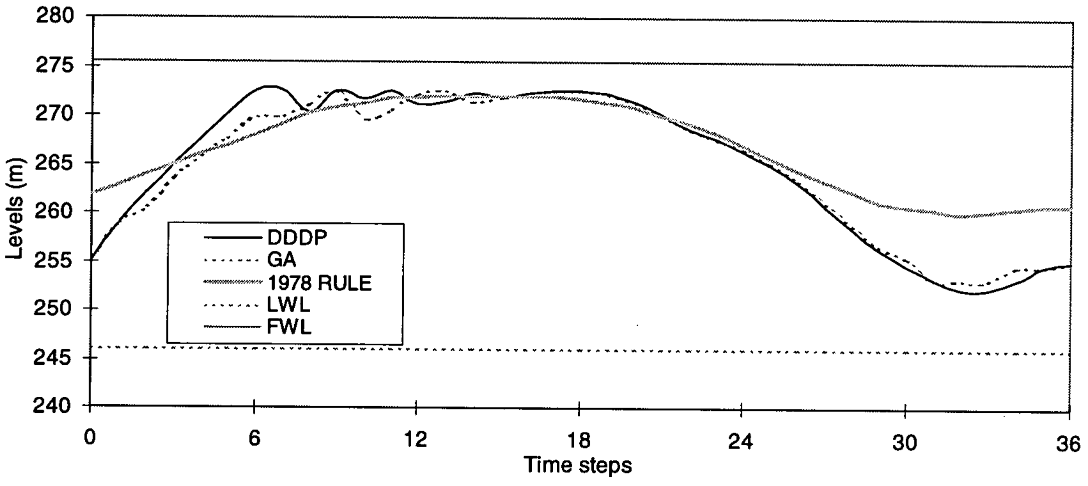


Figure 6.8 Level trajectories, optimising for hydropower and irrigation

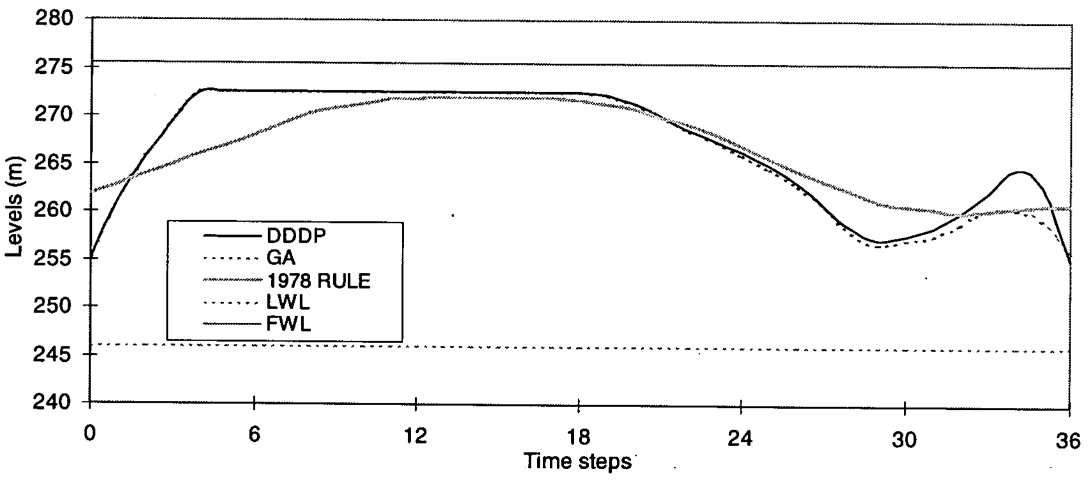


Figure 6.9 Level trajectories, optimising for irrigation only

The objective function values produced by the GA and the DDDP models for the different cases considered are presented in Table 6.2. It can be observed that the GA solutions are very close to the optimum in terms of objective function values achieved. The objective function values achieved by GA were 99.84%, 99.98%, and 99.90 % of those achieved by the DDDP approach for EDS1, EDS2, and EDS3 respectively. The execution time for GA was 90 seconds compared to 50 seconds for DDDP irrespective of the objective function used. In terms of execution time, GA is not as economical for a single reservoir system as the procedure involves a large number of evaluations of the objective function. The difference in the execution time is, however, small and can be considered as insignificant for all practical purposes. Given the vast decision space which the GA had to search for this particular problem, the results can be considered as highly satisfactory. The search space is considered vast because the upper and lower bounds of releases from Karangkates reservoir are 155 and 25 respectively, and the GA had to search for optimal values between these bounds. On the whole, the performance of GA compares well with the DDDP.

Table 6.2 Objective function values achieved by GA

CASE	**GA values as % of DDDP values**
EDS1	99.84
EDS2	99.98
EDS3	99.90

As described in chapter 5, the simulation model developed for the Brantas Basin can be used to evaluate alternative water resource development scenarios in the basin (Wardlaw 1993). A series of economic post processing functions have been incorporated in the model, dealing with agricultural production, hydropower production, aquaculture production, and pumping costs. The model has been used in this study to provide a more detailed evaluation of the operating policies determined by optimisation models. The response of the system to the rule curves produced by the GA and the DDDP was evaluated for all the three cases (EDS1, EDS2, EDS3). The derived rule curves were fed into the simulation model and it was run for a 37-year period of continuous streamflow record.

From the output of the simulation model, the mean deficits in irrigation supply to the Brantas Delta are calculated. The model evaluates deficits in each 10-day time step by satisfying all demands from upstream to downstream and then comparing the irrigation demands of the Brantas Delta with the available river flow. Figure 6.10, Figure 6.11, and Figure 6.12 show the mean deficits in irrigation supplies to the delta for EDS1, EDS2, and EDS3 respectively for the DDDP and the GA derived rule curves, and also for the 1978 rule curve. In this research, a future potable water demand of $8m^3/s$ was considered which resulted in a total demand of 16.5 m^3/s from K. Surabaya. The water resource of the basin is sufficient to permit the continued increase in the potable water supply at the expense of irrigation water supply. It is economic to do so but is totally unacceptable socially. For this reason, the potable demand was given priority over irrigation demands. This resulted in irrigation deficits in dry seasons for each scenario considered but the scale of deficits was different for each scenario. When the optimisation was carried out for maximising irrigation production (EDS3), interestingly both the GA and the DDDP rule curves resulted in slightly higher deficits than EDS1 or EDS2. This is due to the reason that with the rule curve derived in EDS3 the reservoir needs to be maintained at higher levels thereby allowing releases smaller than with the rules derived in EDSI or EDS2. However, in each case considered, significant reduction in irrigation deficits were obtained with the GA and the DDDP derived rule curves over the 1978 rule curve.

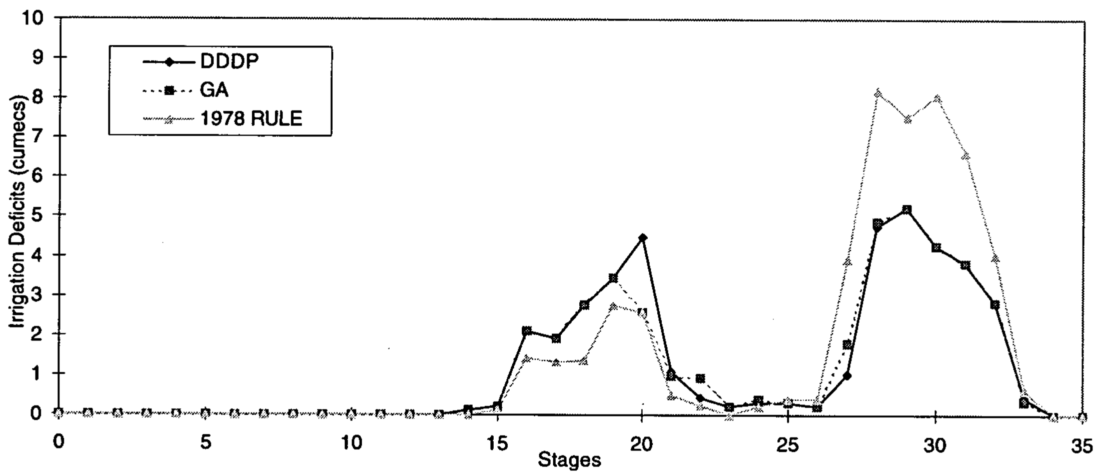


Figure 6.10 Mean irrigation deficits, EDS1

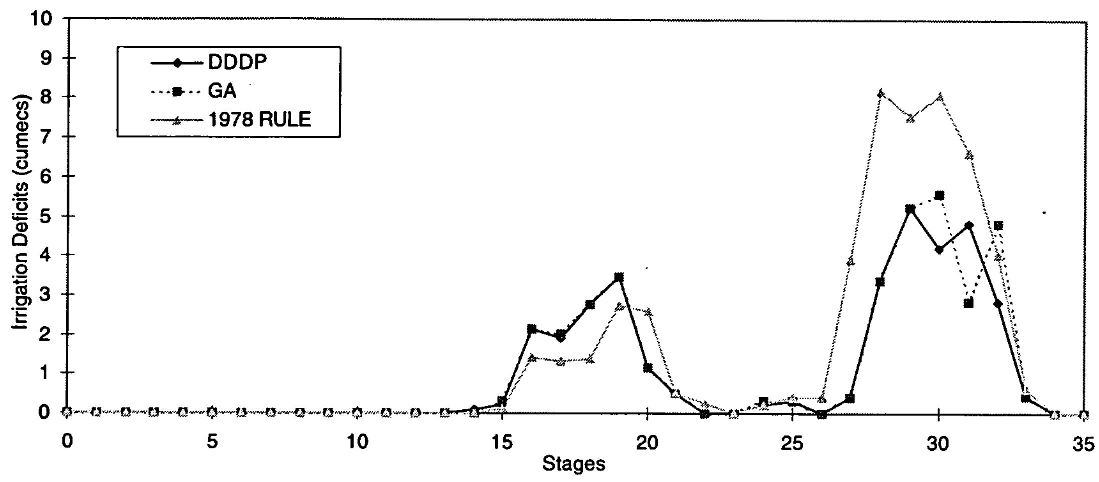


Figure 6.11 Mean irrigation deficits, EDS2

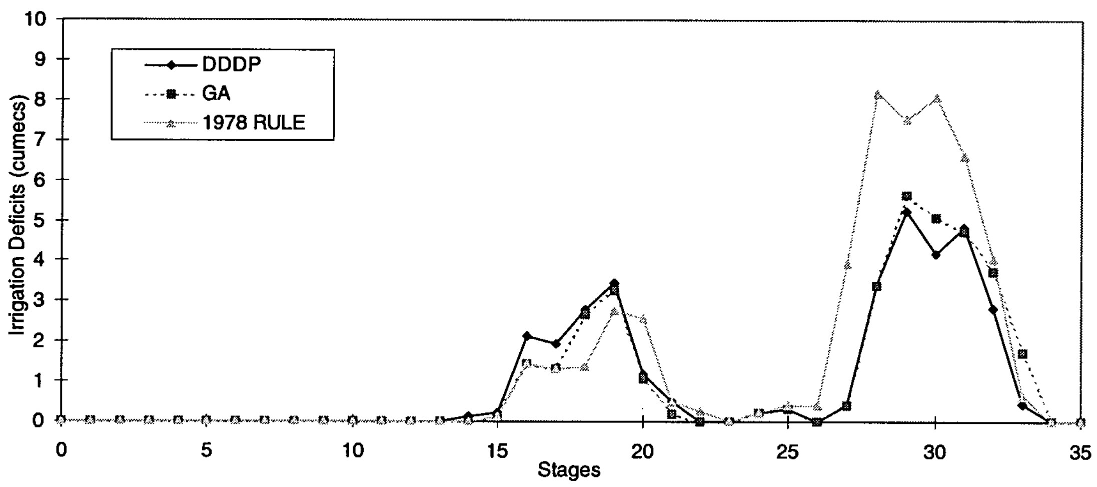


Figure 6.12 Mean irrigation deficits, EDS3

A comparison of the economic returns that can be realised from the basin for each of the scenarios considered is presented in Table 6.3. The model reports results in terms of irrigation production at the Brantas Delta, energy benefits, and capacity benefits at Karangkates among other values. Energy benefits are computed on the basis of the fuel cost of replacing hydropower with thermal power. Capacity benefits relate to difference in capital costs between hydropower and equivalent thermal power station alternatives. The model computes the capacity benefits on the basis of the series of minimum annual 10 day power production figures, and thus give a measure of the reliability of power production. The output from the simulation model includes a range of reliability levels but for economic analysis the mean energy benefits and the capacity benefit at 95% annual reliability or at 99.5% overall reliability, whichever is lower, are considered. The simulation model was also run using the 1978 rule curve, and the economic returns obtained from that run are also shown in Table 6.3 in the last column under the heading RC.

Table 6.3 Economic returns in million Rupiah, existing development situation

	EDS1		**EDS2**		**EDS3**		**RC**
	DDDP	**GA**	**DDDP**	**GA**	**DDDP**	**GA**	
Irrigation production	50681	50905	51626	51591	50929	51002	51104
Karangkates Energy	32981	33058	32741	32789	32867	32846	33321
Karangkates Capacity	3398	3461	3418	3428	3450	3451	3748
Total	87060	87424	87785	87808	87246	87299	88173

The figures in Table 6.3 show that the rule curves derived for the case when optimisation was carried out for both hydropower and irrigation (Case EDS2) produce significantly higher total returns than those obtained when the optimisation was carried out for hydropower only (Case EDS1). For case EDS3 where the optimisation was carried out for maximising the irrigation production only, the total returns obtained from the system are lower than those obtained with EDS1 or with EDS2. The GA derived rule produced higher total returns than the DDDP derived rules for EDS1, EDS2, and EDS3. In all the three cases, however, the differences are small thus indicating that the GA solutions are very close to the DDDP solutions. Of the three different objective functions considered, the best results are achieved when the optimisation was carried for power and irrigation benefits (case EDS2).

With the 1978 rule curve, the return from irrigation production is lower than for the case when the optimisation was carried out both for irrigation and power production (case EDS2) but is higher than the other two cases (EDS1 and EDS3) considered. Although the deficits in irrigation supply with EDS1 and EDS3 are lower than with the 1978 rule, the irrigation production is slightly higher with the 1978 rule. This may be due to the reason that the simulation model evaluates irrigation production on the basis of cropping pattern in the delta, and the crop production functions relate crop yield to water supply through various stages of crop growth. A same amount of deficit in different time steps may lead to different overall value of annual irrigation production. The total return from 1978 rule curve is higher mainly due to the higher power production and capacity benefits. However, it must be emphasised that with the 1978 rule curve higher levels need to be maintained causing large deficits in irrigation supplies particularly during the drier periods. At the same time maintaining the reservoir at higher levels leads to increased power production and increased capacity benefits at the expense of losses in irrigation production. The mean annual irrigation losses amount to 4865 million Rupiah under the 1978 rule curve operation. With the DDDP and GA derived rules in EDS2, the mean annual irrigation losses are 4347 and 4398 million Rupiah respectively.

6.4 First Future Development Scenario 1 (FDS1)

This section presents the application of the GA and the DDDP approach for optimising the combined operation of the Karangkates and the Wonorejo reservoirs for a likely future development scenario. The water resource of the Brantas Basin is almost fully utilised at present but the pressure on the available water resources in the basin is still increasing at a rapid rate due to large scale industrial development and agricultural extension. To optimise the utilisation of available water resources, various development schemes have been proposed (Mott MacDonald 1988). Among those schemes is the construction of Wonorejo dam shown in Figure 6.1. In simplified network terms, FDS1 can be represented as shown in Figure 6.13. Wonorejo is a storage reservoir that can be used for hydropower production and for irrigation. The hydropower production for Wonorejo reservoir can be calculated using the polynomial equations proposed by Nippon Koei (1978) as described in section 6.3.1. One of the purposes of Wonorejo is to supply water to local irrigation schemes. After satisfying these local irrigation demands, the water from Wonorejo can be used to augment the irrigation demands at Brantas Delta. A potable demand of 8 m^3/s has been used which results in a total demand of 16.5 m^3/s from the K. Surabaya, as for the runs with a single reservoir.

To generate a single year of inflow data for Wonorejo, the critical period analysis was carried out on the entire 37 year inflow series in a manner similar to that for Karangkates. The combined operation of Karangkates and Wonorejo reservoirs is optimised over 36 successive 10 day periods using the GA and the DDDP approach for the following objective function.

$$\textit{Maximise } Z = \sum_{i=1}^{n}\sum_{t=1}^{N}\left[E_{it}(\bar{S}_{it}, R_{it}) - P_{it}\right] \tag{6.14}$$

where P_{1t} is the penalty for not satisfying the irrigation demands at delta, and P_{2t} is the penalty for not satisfying local irrigation demands at Wonorejo in time step t.

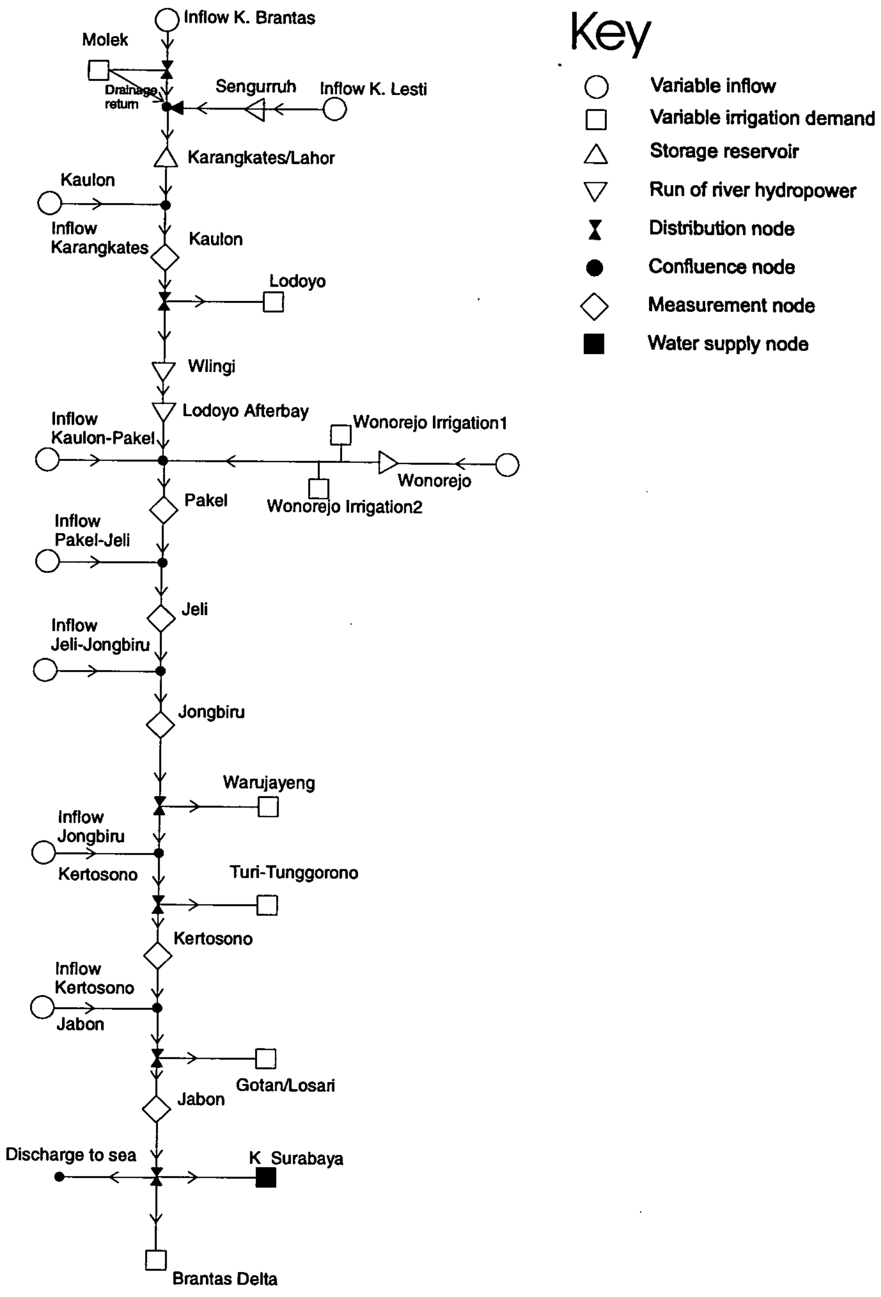


Figure 6.13 Network used in the optimisation models, FDS1

The minimum reservoir level required for generating hydropower at Wonorejo is 143 metres. Only off-peak power could be generated at Wonorejo as there is no re-regulation downstream of it. The rest of the procedure for hydropower computation is same as that for Karangkates. The details of power stations for each reservoir are given in the previous chapter.

6.4.1 The Solution Procedure

With the GA approach, the solution procedure is similar to the one used for a single reservoir system. Since the number of reservoirs included in the model for this future development scenario is two, each chromosome as a potential solution consists of 36 x 2 = 72 genes. The fitness of chromosomes is obtained using an evaluation function which corresponds to the objective function given by (6.14). The same population size of 100 as used for the single reservoir system has been retained, and the model allowed to run for 1500 generations compared to 1000 generations for the single reservoir system. The reason for using higher number of generations is that it is relatively quicker to achieve an optimal sequence of 36 genes compared to 72 genes. The rest of the GA parameters are identical to the ones used for the earlier problems (EDS1, EDS2, and EDS3) except that the mutation probability was changed to 1/72 (0.0138) instead of 1/36.

The manner in which the GA approached the solution is shown in Figure 6.14. There is a steady increase in the maximum and average fitness of the population during the early part of the run but the growth slows down during the later part. Existing solutions improve during the later part of the run but the rate of improvement is slow as compared to the earlier part of the GA run. Variability in average fitness is low indicating that a large number of similar solutions have been identified in the early part of the run.

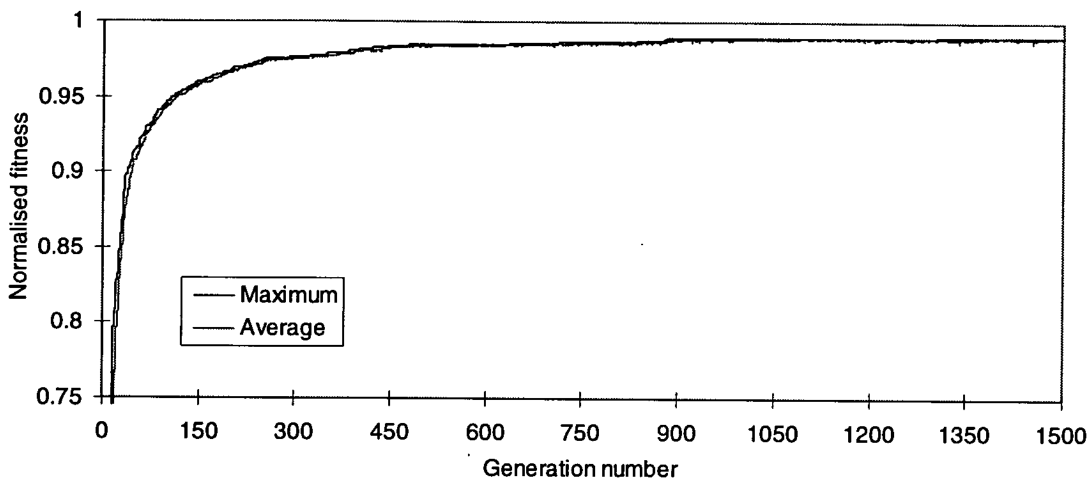


Figure 6.14 Variation of maximum and average fitness with generations, FDS1.

To apply the DDDP approach, the state space must be discretized in a finite number of points. Discretization of decision space is not required as the value of the decision variable can be determined for every feasible pair of states in two consecutive stages. For the development of reservoir states, the total storage volumes of Karangkates was divided into 100 values in increments of 3.535 Mm^3 as before. The storage of Wonorejo was discretized into 50 levels in increments of 2.367 Mm^3. With these numbers of discretization, the total number of possible combinations of states to be tested at each stage of a DP algorithm would be 100^2 x 50^2= 25 million. The application of DP approach to this problem would therefore require huge memory and computing time. For this reason DP cannot be applied to this problem. With DDDP, the number of combinations of states which needs to be tested at each stage is 3^2 x 3^2 = 81 only. The DDDP approach can therefore be used without much computational complexity. The efficiency of the DDDP model however depends largely upon the levels of discretization of the state variables. Any increase in the number of discretizations would increase the number of evaluations of the recursive formula and the computing time increase accordingly. The likelihood of achieving an optimum is also reduced with the increase in the number of discretizations. On the other hand, a coarser discretization may lead to inaccurate results.

The output of the GA model is the optimal reservoir levels for both Karangkates and Wonorejo. The optimal reservoir levels obtained by DDDP and GA for Karangkates have been plotted in Figure 6.15 and the trajectories for Wonorejo have been plotted in Figure 6.16. With GA, the objective function value achieved was 99.07 % of that produced by the DDDP. Both the techniques produce similar objective function values but there is some deviation in the rule curves obtained for both the reservoirs. The DDDP derived rule curve for Karangkates is smoother than the GA derived rule curve. For Wonorejo, the level trajectories produced by both the models match very closely, again indicating the robustness of the approach in identifying near optimal solutions. The execution time for the DDDP was 210 seconds while the GA took 290 seconds to complete 1500 generations.

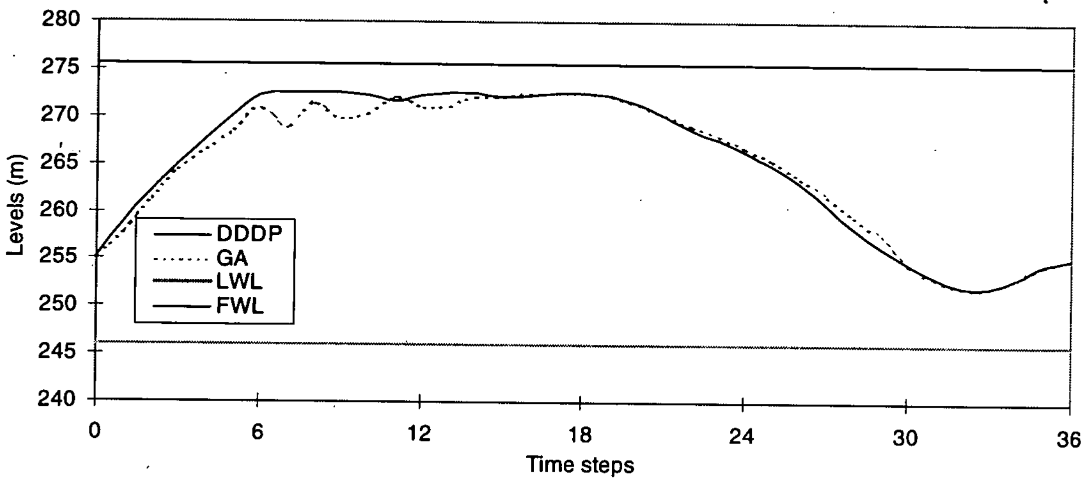


Figure 6.15 Level trajectories for Karangkates reservoir, FDS1

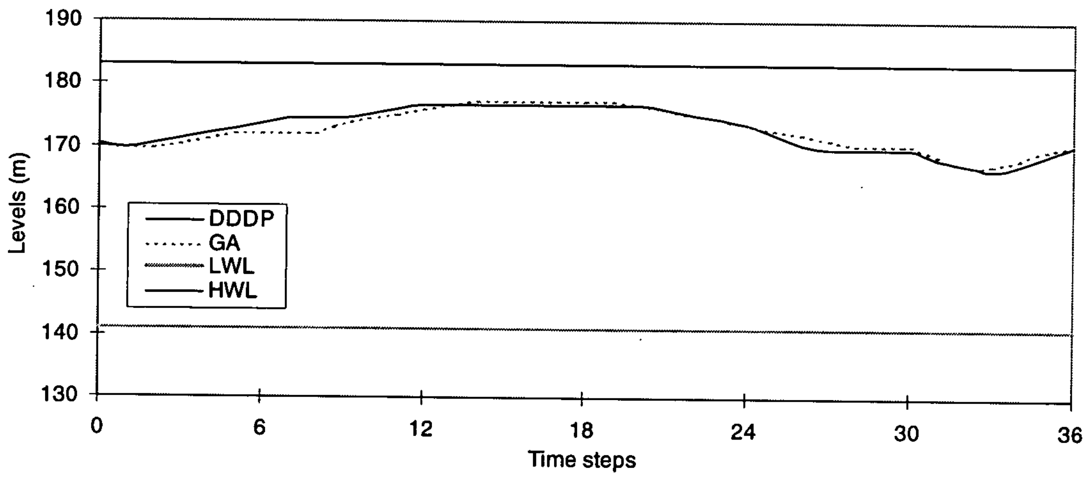


Figure 6.16 Level trajectories for Wonorejo reservoir, FDS1.

Rule curves derived using the optimisation models were fed into the simulation model as in earlier cases. The output of the simulation model was used to calculate the deficits in irrigation supply to the Brantas Delta. Figure 6.17 shows these deficits when Wonorejo was included in the model. It can be observed that there is some decrease in deficits over the scenario when only Karangkates was included in the model. The primary purpose of

Wonorejo is to supply water to local irrigation schemes, and the water is available for irrigation at delta only after satisfying these local demands. It appears that the contribution of Wonorejo to the river flow is not large enough. For this reason, it is not possible to eliminate deficits completely even after including Wonorejo in the system.

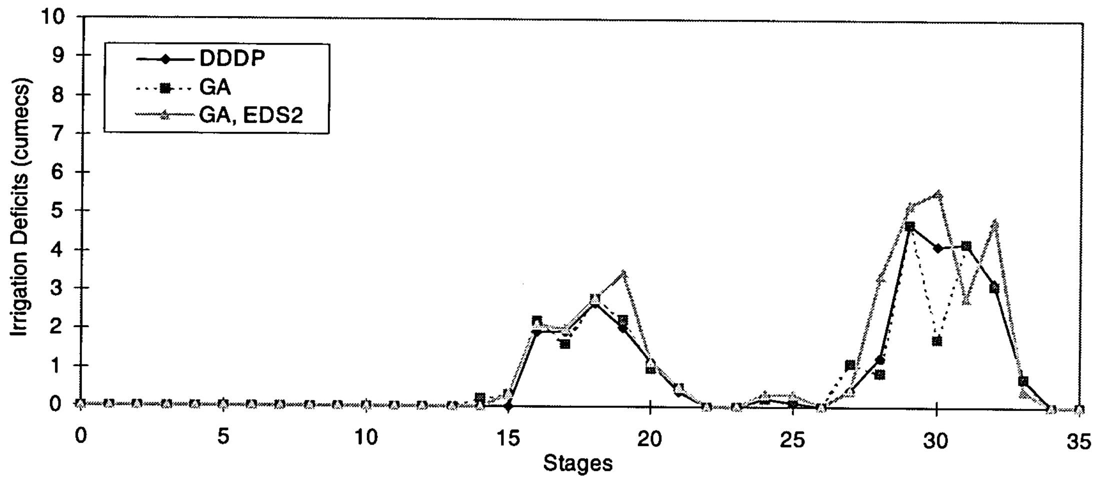


Figure 6.17 Mean irrigation deficits, FDS1

The economic returns from the system are shown in Table 6.4. The values shown in Table 6.4 indicate that the returns increase considerably over EDS1, EDS2 and EDS3 with both the approaches. The increase in irrigation return is realised after an initial capital investment is made in the construction of Wonorejo Dam. Again, the rule curve produced by GA and the DDDP produce very similar total returns with the DDDP based rule curve producing a slightly better overall return from the system. This is largely due to higher returns from the irrigation production at the local schemes fed by Wonorejo.

Table 6.4 Economic returns in million Rupiah, FDS1

	DDDP	GA
Net irrigation production	51935	51939
Wonorejo Irrigation 1	8588	8530
Wonorejo Irrigation 2	5035	4362
Karangkates Energy	32691	32540
Karangkates Capacity	3418	3426
Wonorejo Energy	1481	1472
Wonorejo Capacity	0	0
Total	103148	102269

6.5 Second Future Development Scenario (FDS2)

This section considers the development of operating rules for another future water resource development scenario that includes Beng reservoir in the system. Rule curves have been derived for Karangkates, Wonorejo and Beng reservoir using the GA and the DDDP approach. Beng is a potential offstream reservoir in the lower reaches of the K. Brantas (Figure 6.1). The simplified network representing FDS2 is shown in Figure 6.18. During periods of high flows, water could be pumped back from the main stream to Beng. If the water from Karangkates and Wonorejo are not able to meet the irrigation demands for the delta, then releases could be made from Beng. Two pump stations with a discharge capacity of around 10m^3/s would be used to divert Brantas flows to Beng in the wet season, in addition to the local inflow to the reservoir. The volume of pumping in each time step is determined on the basis of the naturalised available Perning discharge (QAP). There are two major demands downstream of Perning, irrigation demand at delta and potable water demand for the City of Surabaya, that need to be satisfied before any water is diverted to Beng. For this reason, a threshold value of 70 m^3/s has been assumed. If QAP is less than this value, no water is pumped to Beng. If QAP is greater than the threshold value, the surplus amount is pumped to Beng subject to the constraints on the maximum rate of pumping.

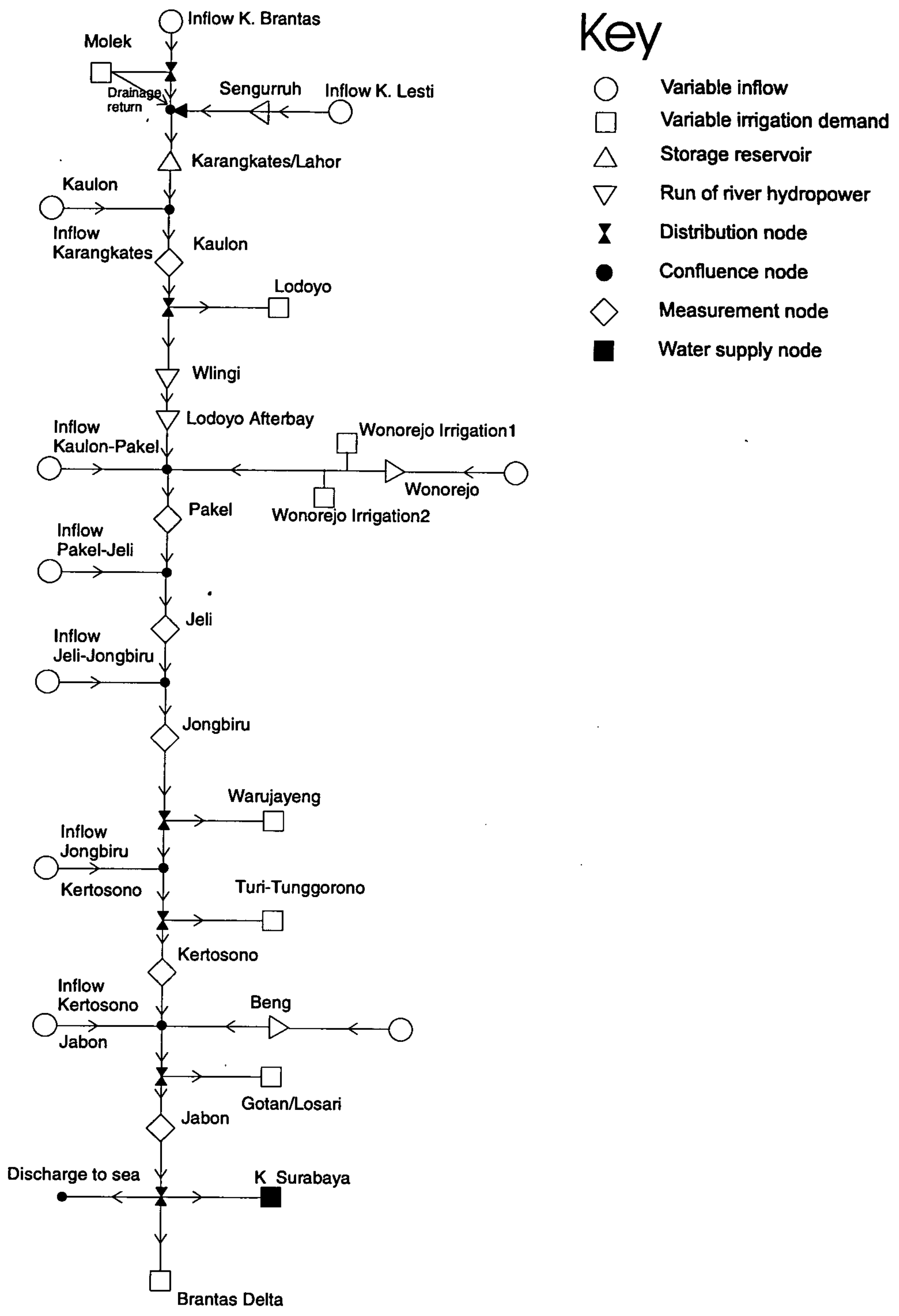


Figure 6.18 Network used in optimisation models, FDS2

There is a provision of hydropower production at Beng, and it has been assumed that the cost of pumping will be offset by the power production. A minimum level of 55.0 m has been assumed for power production. If the water level falls below this value, no power could be produced. As for Wonorejo, only offpeak power could be generated at Beng as there is no re-regulation downstream of the reservoir. The objective function is modified only to reflect an additional reservoir in the system, and may be expressed as

$$Maximize\, Z = \sum_{i=1}^{n} \sum_{t=1}^{N} \left[E_{it}(\overline{S}_{it}, R_{it}) - P_{it} \right] \tag{6.15}$$

where P_{1t} is the penalty for not meeting the local irrigation demands at Wonorejo, P_{2t} is penalty for not meeting the irrigation demand for the delta, and P_{3t} is zero.

6.5.1 The Solution Procedure

The chromosome required to represent the solution for this case comprises 108 genes (3 reservoirs, 36 time-steps). A higher population size of 200 has been used to maintain greater diversity in the population as the problem under consideration is more complex in the sense that the optimal sequence of 108 genes is required compared to 36 and 72 genes for the earlier problems. The model was allowed to run for 2000 generations. A crossover probability of 0.7 and a mutation probability of 0.009 (1/108) was used. The GA code used remains unchanged from the one used for the problems discussed earlier. The modifications are required only to the evaluation function and the data file which may be different for different objective functions. Changes in the evaluation function require no changes to other modules of a GA. The ease with which different problems can be solved using the same code is perhaps the most important advantage of the GA. The rest of the solution procedure is same as that for other problems discussed earlier.

Figure 6.19 shows the progress of maximum and average fitness over 2000 generations. As is typical of GA performance, the rate of improvement of solutions is quite high in the beginning of the run. However, as the GA run progresses, the population starts becoming matured and the rate of improvement becomes steady. Slight improvements could be observed in the existing solutions but no large changes are visible. The variability in average fitness is also low.

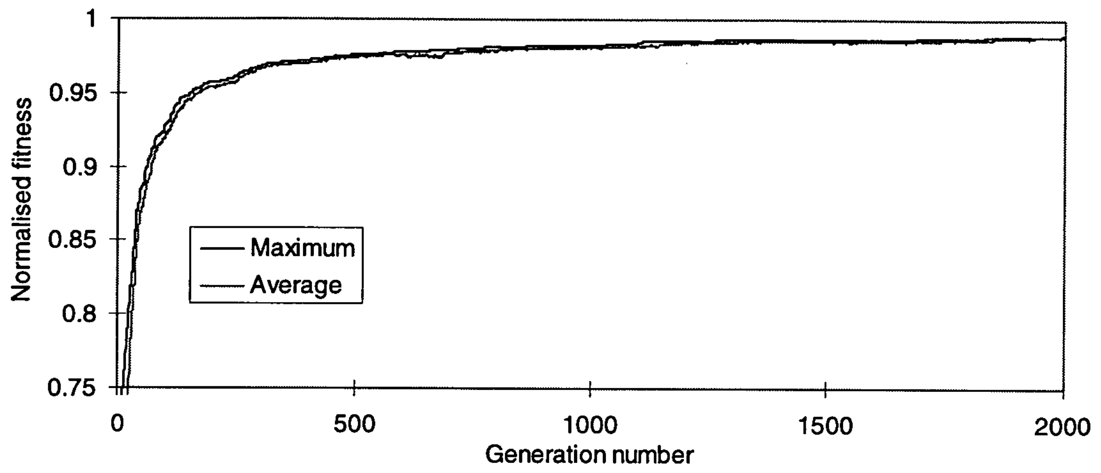


Figure 6.19 Variation of maximum and average fitness with generations, FDS2

A DDDP model for the three reservoir system including Karangkates, Wonorejo and Beng was also developed. To apply the DDDP approach, the storage of each reservoir was discretized in a finite number of values as in earlier cases. The storage of Karangkates reservoir is discretized into 100 levels in increments of 3.535 Mm^3 while the storage of Wonorejo and Beng is discretized into 50 values each in increments of 2.367 Mm^3 and 3.23 Mm^3 respectively. It took around 425 seconds for the DDDP procedure to complete a run on a Pentium based 586 PC. With GA, the time required to complete 2000 generations was 590 seconds. For the problem solved by Esat and Hall (1994), the computational effort required to solve the problem using DDDP increased exponentially. Fahmy et al. (1994) reported that the execution time for a GA increased at a significantly slower rate than with the DP. In this study, the execution time for a GA has gone up in a similar proportion to the DDDP model. However, with the increase in the dimensionality of system, DDDP becomes computationally bounded but GA can still be used for achieving acceptable solutions to such problems. This was demonstrated in chapter 4 where a ten reservoir problem was solved within an affordable computing time. An additional drawback of the DDDP approach is that the computer code has to be modified for each problem considered.

The optimal trajectories produced by GA and by DDDP for the three reservoir system are shown in Figure 6.20, Figure 6.21, and Figure 6.22. The trajectories for Karangkates appear to be in very good agreement with the DDDP trajectories for the second half of the operating horizon but there are some small deviations in the first half. Although this being the case, both the trajectories represent solutions within 1.1% of the optimum. The objective function value produced by GA was 98.9% of that produced by the DDDP. For Wonorejo, the trajectories match closely in all time steps but there are deviation for the trajectories for Beng during the later part of the operating horizon.

The differences in trajectories for Beng could be due to the reason that the objective function considered here is a function of releases made from the reservoir as well as the water head in the reservoir, and the same amount of power can be produced by different combinations of releases and heads. For example, maintaining the reservoir at a higher level and releasing less amount of water can result in the same power that can be produced by maintaining the reservoir at a lower level and releasing higher amounts. Therefore, the trajectories while looking different may still represent similar solution.

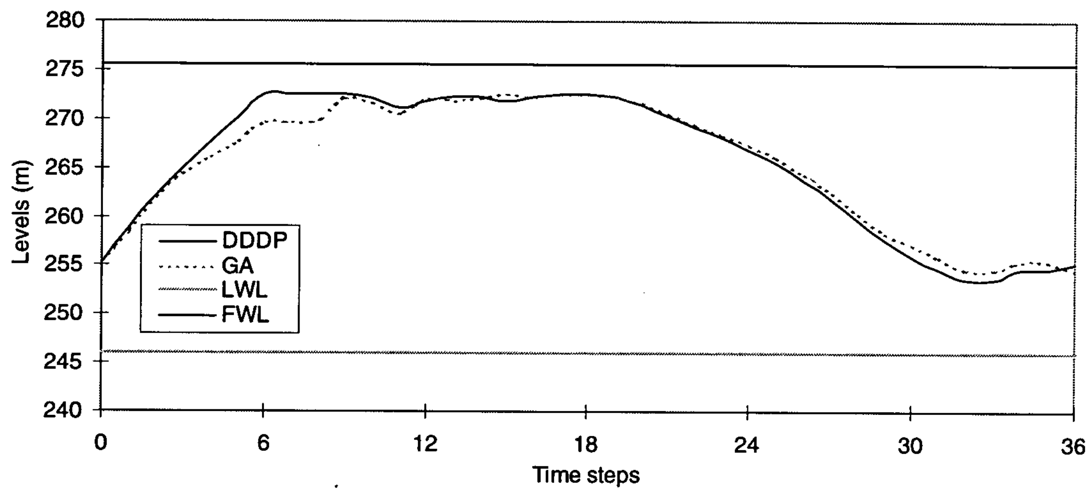


Figure 6.20 Level trajectories for Karangkates, FDS2

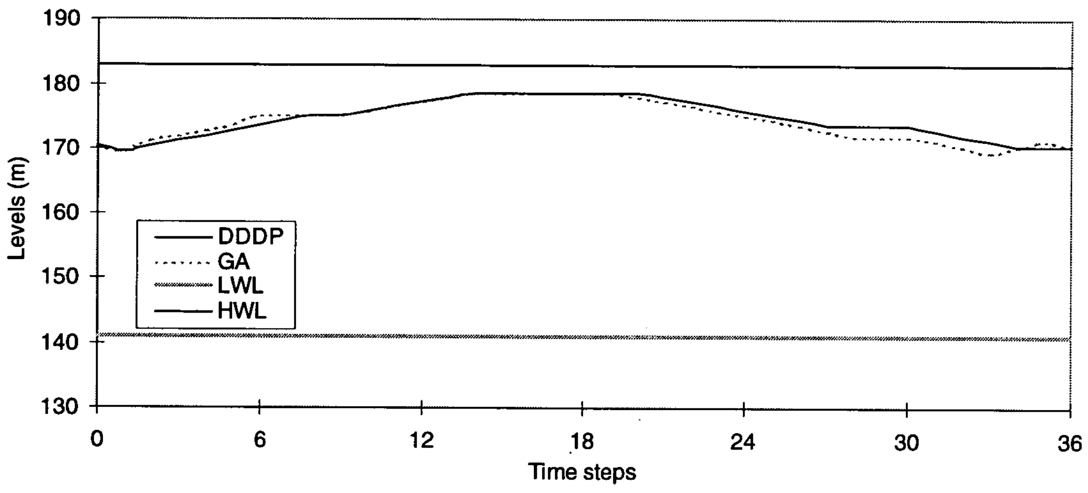


Figure 6.21 Level trajectories for Wonorejo, FDS2

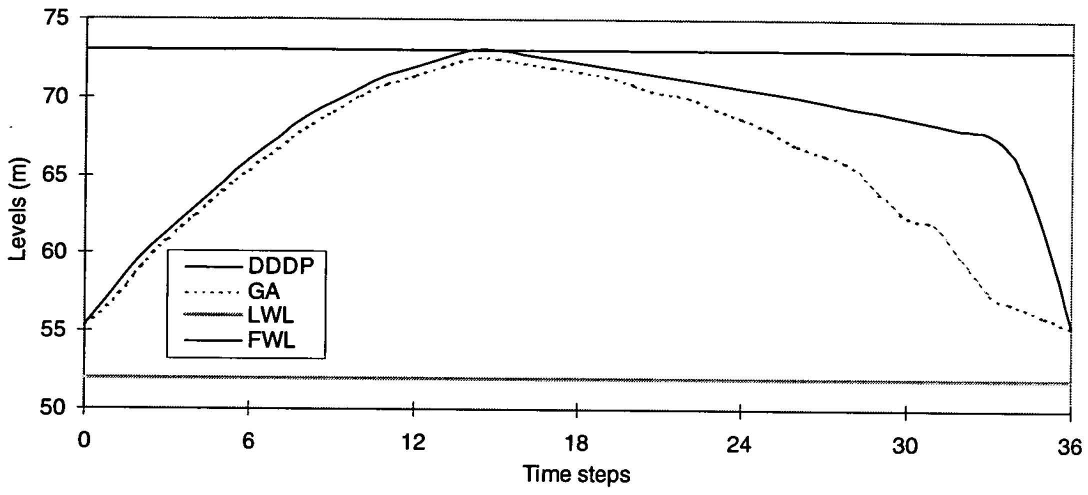


Figure 6.22 Level trajectories for Beng, FDS2

The simulation model was run with the rule curves derived using the optimisation models. The mean deficits in irrigation supply to the delta are shown in Figure 6.23. It can be seen that with the inclusion of Beng in the system, there is a further decrease in the mean deficits but it was not possible to eliminate the deficits totally. The main reason these deficits occur is that the historic inflow set includes several years characterised by lower inflow sequences than that used in the derivation of rule curves. It can be observed from Figure 6.2 that the independent inflow series are more extreme than the one used in the optimisation models to derive the rule curves. Additionally, it appears that the inclusion of Beng in the system does not sufficiently increase the irrigation supply to the delta in extremely dry years to have any significant impact on the deficits during these years.

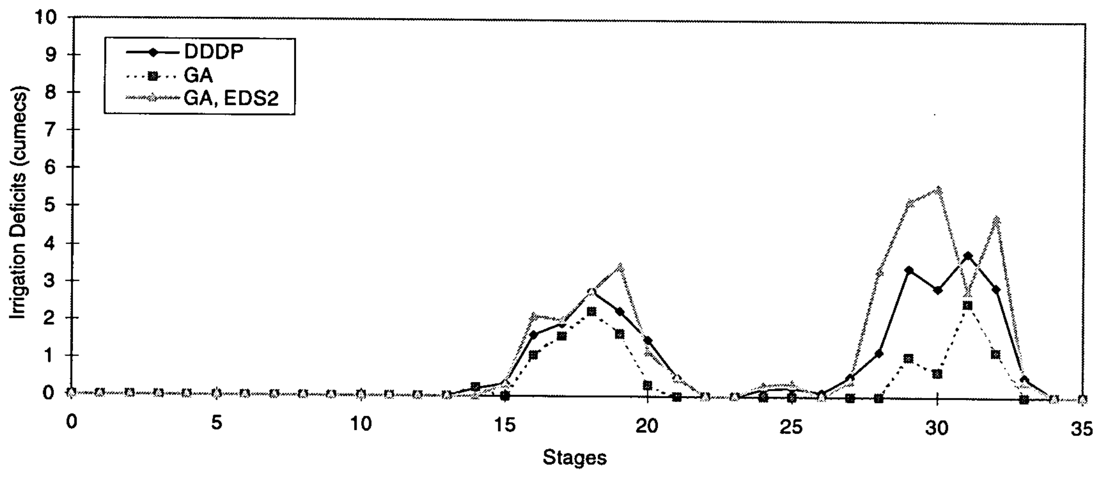


Figure 6.23 Mean irrigation deficits, FDS2

Table 6.5 presents the economic returns from the operation of the reservoir system under FDS2. The total return from the system for the GA based rule is higher than that produced by the DDDP, which is highly satisfactory given the complexity and the size of the problem considered. The main reason for this increase is the higher returns with the GA derived rule curve from irrigation production at the Brantas Delta. Also, the capacity benefits at Karangkates are higher with the GA derived rule curve.

Table 6.5 Economic returns in million Rupiah, FDS2

	DDDP	GA
Net irrigation production	52045	52527
Wonorejo Irrigation 1	8648	8468
Wonorejo Irrigation 2	4656	4600
Karangkates Energy	32657	32821
Karangkates Capacity	3419	3498
Wonorejo Energy	1510	1510
Wonorejo Capacity	0	0
Beng Energy	1141	1137
Beng Capacity	0	0
Total	104076	104561

6.6 Sensitivity To Mutation Probability

Proper settings of GA parameters are required to achieve the best performance from it. Therefore, a sensitivity analysis has been carried out to determine the effect of crossover probability and mutation probability on the GA performance. To analyse the effect of mutation probability on the GA performance, a crossover probability of 0.7 was used, with all other parameters and run controls as discussed earlier.

Sensitivity of the GA performance to mutation probability is presented in Figure 6.24. Fitness is expressed as the proportion of the optimum produced by the DDDP. It can be observed from the results of sensitivity analysis that although the chromosome length is different for each case considered in this study, the mutation probabilities that yield good performance are those that corresponding to a single mutation per chromosome. The performance of the objective function with respect to mutation is similar to that for the four-reservoir problem solved in chapter 4. It is interesting to note that in all cases, 1 mutation per chromosome produces about the best results. For a single reservoir system, the best results were obtained with a mutation probability of 0.027 (1/36) for EDS1 and EDS3 and 0.033 (1.2/36) for EDS2 respectively. For the two-reservoir case (FDS1), the best mutation probability is 0.0139 (1/72) while for the three reservoir system, it is 0.007 (0.8/108). The optimum number of mutations per chromosome in each case was around 1. Since EDS1, EDS2, and EDS3 use the same number of genes to represent the problem, it was considered appropriate to show the results of sensitivity analysis for EDS2 only. The trend of results for EDS1 and EDS3 were very similar to EDS3.

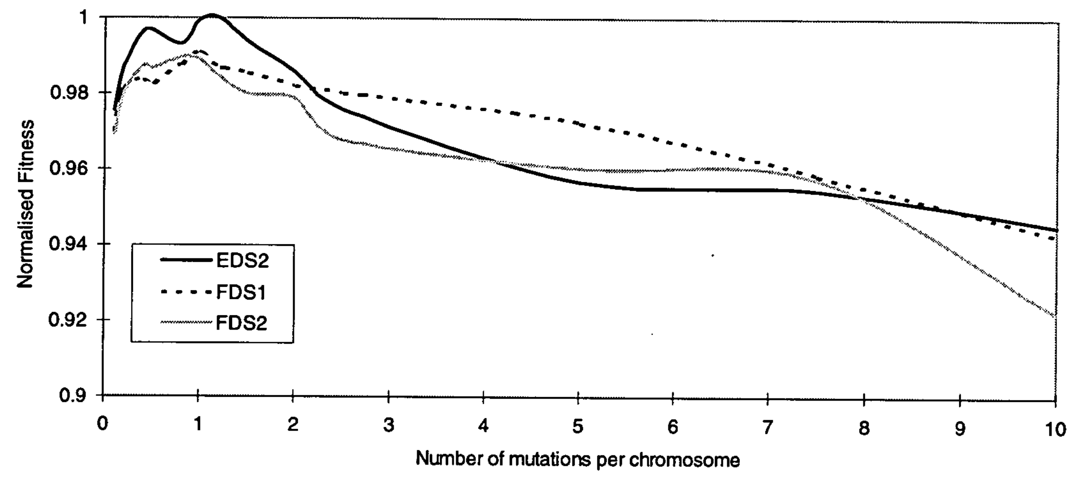


Figure 6.24 Effect of mutation probability on GA performance for different cases

The analysis showed that with the increase in the number of mutations per chromosome, GA performance deteriorates. This might be due to the fact that causing too many disturbances to already good solutions deflects them from optimal paths. On the other hand, causing only a few changes tends to improve the existing solutions. Clearly, more than two mutations does not lead to good GA performance. The performance of the objective function with respect to mutation is similar to that for the four reservoir problem solved in chapter 4.

6.7 Sensitivity To Crossover Probability

The effect of crossover probability on the performance of GA has also been analysed using a mutation probability which corresponds to 1 mutation per chromosome irrespective of the length of the chromosome. The mutation probabilities were therefore set at 0.027, 0.0138, and 0.009 for the single reservoir, two reservoir and the three reservoir case respectively. GA runs were carried out with crossover probabilities ranging from 0.5 to 1.0. Figure 6.25 shows the sensitivity of the achieved fitness to crossover probability for each of the three cases. Again, the fitness has been expressed as the proportion of the optimum produced by the DDDP

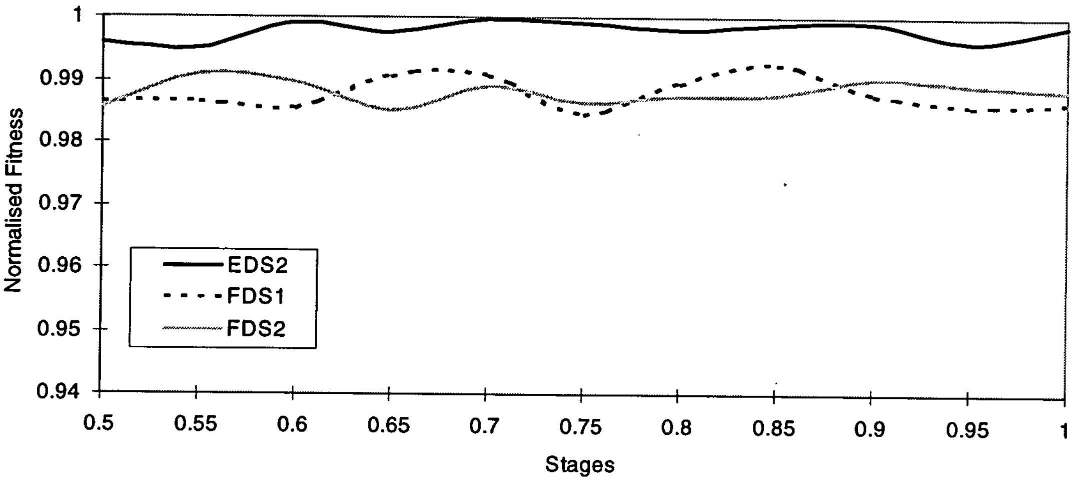


Figure 6.25 Effect of crossover probability on GA performance for different cases

The performance of GA is stable with respect to crossover probability. The variability of normalised fitness with crossover probability is much lower than that with mutation probability. The trend of results is similar for all the problems considered here and for the four reservoir problem solved earlier in chapter 4. The results of sensitivity analysis for mutation and crossover probabilities have shown that very good results could be obtained for some particular combination of crossover and mutation probability but most combinations produce satisfactory results. It can be concluded that the impact of crossover probability on the performance is insignificant but definitely some care should be exercised while selecting the number of mutations to be made to each chromosome.

6.8 Evaluation of Economic Post Processing Results

This section presents a comparison of economic returns that could be realised under different scenarios in the basin. The response of the system to the rule curves produced by the GA and the DDDP was evaluated for all the five cases (EDS1, EDS2, EDS3, FDS1, FDS2). The derived rule curves were fed into the model and simulation was carried out for a 37-year period of continuous streamflow record. A comparison of the total economic returns that can be realised from the basin for each of the scenarios considered is presented in Figure 6.26.

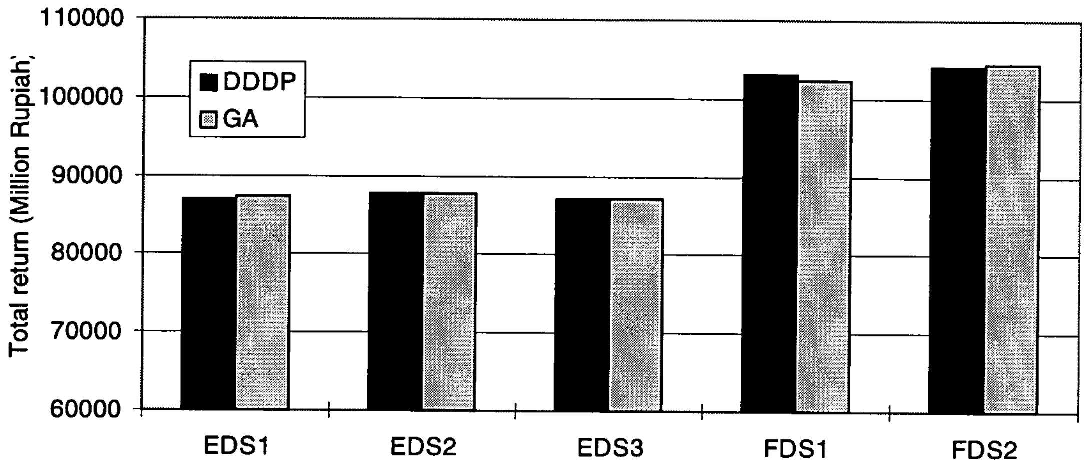


Figure 6.26 Summary of economic returns in million Rupiah under different scenarios

From the results presented in this chapter, it can be observed that in all the cases considered, the GA was able to locate solutions within 1.1% of those produced by DDDP. In four of the five cases considered (EDS1, EDS2, EDS3, and FDS2) in this research, the total economic returns for the system are higher with GA derived rules than with the DDDP derived rules. In the remaining case, the total economic return produced by the GA was sub-optimal by a very small amount (0.85%). The results thus demonstrate that the GAs can be effectively used in identifying optimal operating policies for multi reservoir systems.

6.9 Conclusions

A real world reservoir operation problem has been solved satisfactorily using the GA technique. The existing development situation in the Brantas Basin was considered with different objective functions. The rule curves derived for EDS2 achieved better results than the original 1978 rule curve in terms of irrigation production, which indicates that the operation of Karangkates reservoir can be improved by its application. Two future water resource development scenarios in the basin were also considered. It was found that the irrigation demand of the basin could be met more often when the Wonorejo and the Beng reservoirs are included in the system. The agricultural production would increase when all the three storage reservoirs of the basin are in operation, and the overall economic returns from the system would also be higher.

The results of GA were compared with DDDP and it was observed that GA was able to locate solutions very close to the optimum for all the cases considered. The objective function values achieved by GA were within 1.1% of that obtained by DDDP. In terms of execution time, the performance of DDDP was better than the GA but the differences are negligibly small. Significant savings in computing time and memory requirements could be obtained for large reservoir systems. The requirements in terms of memory and execution time are less for DDDP than GA when applied to small problems. With the increase in the dimensionality of the system, GAs become economical in terms of memory requirements and execution time.

The results of research presented in this chapter indicate that GAs do not suffer from problems of dimensionality and it is possible to achieve near optimal solutions to multi reservoir systems within the available computing resources. Although, the benefits achieved may be sub-optimal, no existing method can provide truly optimal policies for large real world multi reservoir operation problems. The method does not require discretization of state and decision variables and is equally applicable for non-differentiable and discontinuous functions The ease with which GA could be implemented is also remarkable. A general GA code as the one shown in appendix B could be used to solve different problems with changes required only to the evaluation module and the data file. This is not possible with DDDP. It may be concluded that the employment of GAs in the planning and analysis of water resources systems is showing considerable potential for the future.

7. REAL TIME OPERATION OF THE EQUATORIAL LAKES SYSTEM

7.1 Introduction

This chapter discusses the application of the GA technique in developing real time water management policies for the Equatorial Lakes system on the River Nile in Africa. The rapid increase in the population and the water demand forecasts of the Nilotic countries has created the need to explore and implement water conservation policies for the Equatorial Lakes system. The demand for electricity is increasing and there is an urgent need to develop strategies for generation of hydroelectric power at reliable levels (Wardlaw 1998).

Owen Falls dam is a hydropower facility located at the source of the Victoria Nile at Lake Victoria. It has an installed capacity of 180 MW. An extension is currently under construction that will increase the installed capacity to 260 MW. The potential benefits of the extension cannot be fully realised unless efficient operating policies are developed for Lake Victoria. The objective of this application is to investigate operating procedures for the Equatorial Lakes system that could increase the reliability of power production from the system now and under future development conditions.

Following this introduction, section 7.2 describes the physical characteristics of the Equatorial Lakes system. Section 7.3 describes the development and essential features of the simulation model of the system. Results of simulation model runs are presented in section 7.4. Stochastic flow forecasting models are described in section 7.5. Section 7.6 describes simulation model runs carried out using a synthetic sequence. Real time operation of a single lake system and a three lake system using the GA and the DDDP models is discussed in section 7.7 A power production forecasting approach using GA is presented in section 7.8. Conclusions are presented in section 7.9.

7.2 The Equatorial Lakes system

Lakes Victoria, Kyoga and Albert constitute the Equatorial Lakes system (Figure 7.1). The Equatorial Lakes are natural reservoirs with vast storage capacities. Lake Victoria is the second largest fresh water lake in the world. It has a surface area of 69000 km^2 and a storage capacity of 3121 billion cubic metres (BCM) at an elevation of 1136.3 m. The Lake Victoria basin covers an area of 194000 km^2, of which 44000 km^2 lies in Kenya, 84000 km^2 in Tanzania, 32000 km^2 in Uganda and 33600 km^2 in Rwanda and Burundi. The Lake plays an important role in the fisheries, hydropower, recreation, water supply and transport sectors of the region.

The two major lakes downstream of Lake Victoria are Lake Kyoga and Lake Albert. Lake Kyoga is relatively much smaller in volume terms than Lake Victoria with a storage capacity of 20 BCM and a surface area of 4800 km^2. Lake Albert has a surface area of 6000 km^2 and a storage capacity of 176 BCM. Table 7.1 provides information on historical minimum and maximum lake levels and discharges from the three lakes. These levels are relative to gauge datums of 1122.887 m, 1020.61 m, and 609.82 m for Lakes Victoria, Kyoga, and Albert respectively.

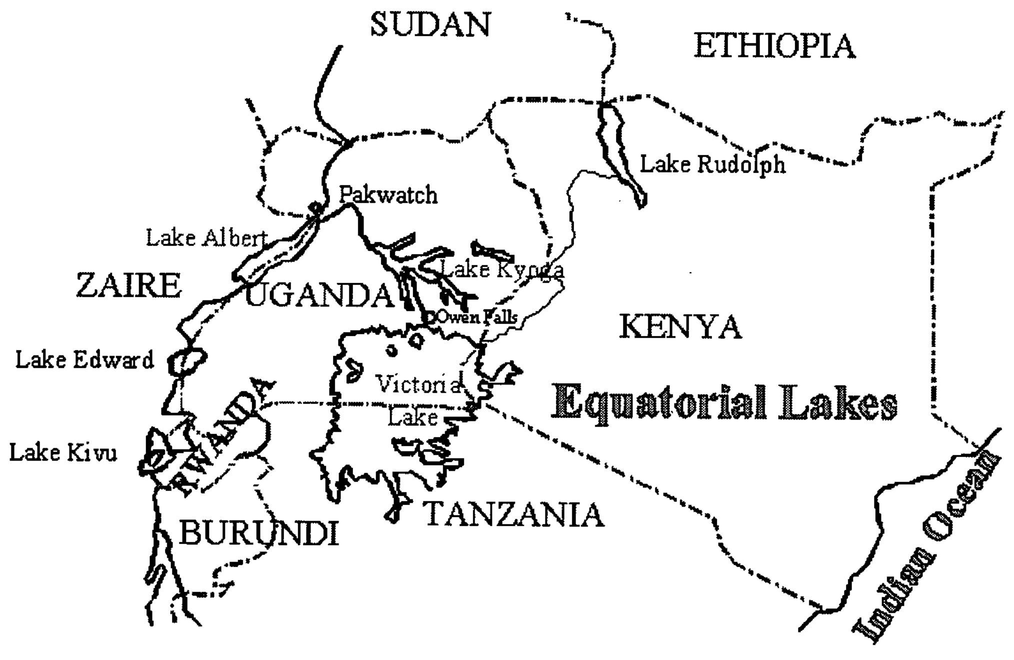


Figure 7.1 The Equatorial Lakes System

Table 7.1 Historical levels and outflows

Lakes	Gauge Levels (m)		Outflows (m^3/s)	
	Minimum	Maximum	Minimum	Maximum
Victoria	10.22	13.33	345.0	1722.0
Kyoga	9.53	13.53	280.1	1990.7
Albert	9.11	14.14	383.9	2120.4

7.3 The Simulation Model

7.3.1 General

A simulation model has been developed by Wardlaw (1998) to assess the performance of the Equatorial Lakes system under different regulation plans. The objective of regulation is to increase the reliability of power production from Lake Victoria as hydropower at Owen Falls, the source of White Nile, accounts for 90% of total installed capacity in Uganda. The model synthesises the monthly water balances of Lakes Victoria, Kyoga and Albert under the influences of different operational controls at Owen Falls dam on Lake Victoria. There is provision for computation of hydropower at Owen Falls, and at the proposed new hydropower facilities at Owen Falls Extension, Bujagali, Kalagala, Murchison, Kamdini and Ayago. The location of Owen Falls is shown in Figure 7.1. The purpose of the model is to evaluate the impact that the various operational controls would have had on historical water levels and discharges had they been in place and experienced the historical record. The basis of analysis is that the hydrological past, as represented in the historical records, is representative of likely future conditions.

The simulation of all lakes is driven by the net basin supply (NBS), defined as inflow minus net evaporation from the lake. The model computes the net basin supply, $Q_{nbs,v}$ from observed outflows and water levels using the following equation.

$$Q_{nbs,v} = O - \Delta S \qquad (7.1)$$

where O is the observed outflow in time step t and ΔS is the change in storage between time steps t and $t+1$ computed from end of period water levels.

7.3.2 Model Data Base

7.3.2.1 Lake Victoria

The water levels recorded at Jinga gauge station, and outflows observed at Owen Falls are used to compute the NBS to Lake Victoria. This historical data has been extracted by Wardlaw (1998) from the reports by the Institute of Hydrology (1993), and by Kennedy and Donkin (1996), and is available for the period between 1899 to 1997. Recent years of record were obtained directly from the Directorate of Water Development (DWD) in Uganda. The elevation-storage characteristics have been taken from Georgakakos (1996).

7.3.2.2 Lake Kyoga

The simulation of Lake Kyoga is driven by outflows from Lake Victoria, and local NBS. It has been more difficult to define the NBS to Lake Kyoga than to Lake Victoria because of limitations in data availability. The NBS record has again been constructed by Wardlaw (1998). The outflow data has been taken from Kennedy and Donkin (1996) for the period 1899 to 1995. A combination of methods had been used to construct this record. The basis of the record is Kyoga outflows at Kamdini from 1940 to 1980, Masindi Port flows from 1912 to 1939, and regression with the Jinja gauge for the periods between 1896 and 1912, and between 1980 and 1995. To compute the NBS, water levels are also required for the period from 1899 - 1997 but the end of month gauge levels at Masindi Port are available only for the period 1915 to 1977. To compute the outflows for the rest of the period, the following relationship has been derived using the common period of Masindi Port levels and Kyoga outflows.

$$Q_k = 0.00067352\, H^{5.717553} \qquad (7.2)$$

where Q_k is the Lake Kyoga outflows in m^3/s and H is the Masindi port water level (gauge level). The local NBS for Lake Kyoga, $Q_{nbs,k}$ is computed as follows.

$$Q_{nbs,k} = Q_k - \Delta S - Q_{outflow,v} \qquad (7.3)$$

where $Q_{outflow,v}$ is the outflow from Lake Victoria and ΔS is the change in storage at Lake Kyoga between consecutive time steps. The changes in storages are computed from Masindi Port water levels, using an elevation-storage relationship given by Georgakakos (1996).

7.3.2.3 Lake Albert

The simulation of Lake Albert is also based upon the local NBS and inflows received from Lake Kyoga. Water levels are recorded at Butiaba on the eastern shore of the Lake, and discharge have been recorded on the Nile downstream of the Lake at Panyango. The water level records at Butiaba are reasonably complete between 1948 and 1976, but thereafter become intermittent. The records of discharges at Panyango on the Albert Nile are also reasonably complete between 1948 and 1979. These records have been extracted from Shahin (1985). On the basis of available records, the following rating relationships have been derived by Wardlaw (1998).

$$Q_A = 0.3848(H-2.0)^{3.5173} \qquad H < 10.55 \qquad (7.4)$$

$$Q_A = 690.4(H-9.5)^{0.8662} \qquad 10.55 \le H \le 12.33 \qquad (7.5)$$

$$Q_A = 977.9(H-9.5)^{0.5127} \qquad H > 12.33 \qquad (7.6)$$

where Q_A is Lake Albert outflow in m^3/s and H is Butiaba gauge level in metres.

7.3.3 Power Components

A master plan for hydropower development in Uganda has been completed recently by Kennedy and Donkin (1996). The master plan covers hydropower potential on the River Nile, and on other smaller rivers in the country. The simulation model (Wardlaw 1998) has provision for hydropower computations at the existing Owen Falls power station, Owen Falls Extension, and the proposed power stations at Bujagali, Kalagala, Murchison, Kamdini and Ayago. The characteristics of different hydropower stations as represented in the model are described below.

7.3.3.1 The Existing Owen Falls Power Station

This power station is located at the source of the Victoria Nile and house 10 turbines of 18 MW rating (recently upgraded from 15 MW), thus giving a total installed capacity of 180 MW. The maximum discharge through each turbine is of the order of 105 m^3/s, and with 9 units in service the total discharge is around 950 m^3/s. Due to the non availability of efficiency and the hydraulic loss characteristics of the station, the power production has been evaluated using the following relationship.

$$P = 0.17045 \times Q \tag{7.7}$$

where P is the power output in MW and Q is the turbine discharge in m^3/s. The above relationship reproduces the power output given by Kennedy and Donkin (1996).

7.3.3.2 Owen Falls Extension

This hydropower station is still under construction. It has a provision for 5 turbines of 40 MW rating but at this stage only two turbines are being installed. On completion of the extension, the installed capacity at Owen Falls will be 242 MW and the corresponding maximum discharge will be of the order of 1420 m^3/s. For the computation of hydropower, the same relationship as that for existing power station has been used. It has also been assumed that the power station is currently operational.

7.3.3.3 Bujagali Falls

There is little storage available at Bujagali, and on the basis of data presented by Kennedy and Donkin (1996), the following relationship has been derived for net head (Wardlaw 1998).

$$H_n = 17.0 - (Q - 660.0) / 722.0 \tag{7.8}$$

where H_n = turbine net head (m), and Q = turbine discharge (m^3/s).

The power production is given by

$$P = 0.00873\, Q H_n \tag{7.9}$$

where P is the power output in MW. There will be four turbine units of 45 MW capacity each at Bujagali.

7.3.3.4 Kalagala Falls

Kennedy and Donkins (1996) presented a number of options for Kalagala Falls power station. For this study, the 315 MW option has been selected. The following net head relationship has been derived for the Kalagala Falls by Wardlaw (1998).

$$H_n = 28.3 - (Q - 660) / 684.0 \qquad (7.10)$$

The same power equation as that for Bujagali may be used. The total installed capacity will be 315 MW (7 units of 45 MW capacity each).

7.3.3.5 Murchison Falls

A number of options for Murchison Falls power station were presented by Kennedy and Donkin (1996). Option M4 has been selected for this study. From the data presented by Kennedy and Donkin (1996), a net head of 87.0 m has been assumed. The power relationship remains the same as that for Bujagali. There will be 6 turbine units of 52.5 MW at Murchison Falls.

7.3.3.6 Kamdini Falls

A number of options have been presented by Kennedy and Donkin (1996) for Kamdini Falls power station. Option KM4 has been selected for inclusion in the simulation model. For this option there will be 3 turbine units of 45 MW capacity each. From the data presented by Kennedy and Donkin (1996), a net head of 25.3 m has been assumed by Wardlaw (1998). The efficiency for energy production and the power relationships are assumed to be same as at Bujagali.

7.3.3.7 Ayago Falls

Kennedy and Donkin (1996) presented a number of options for Ayago Falls power station. The option with 6 turbine units of 38 MW capacity has been included in the simulation model. A net head of 52.0 m has been assumed, and the efficiency and power relationships are assumed to be same as at Bujagali (Wardlaw 1998).

7.3.4 Net Basin Supply Variability

The simulation model of the Equatorial Lakes system is driven by the NBS for each lake. Figure 7.2 shows the percentage deviation of NBS from the long term historical mean for Lake Victoria. There is large variability in NBS with very low NBS experienced, for example, during early 1920's, and very high NBS during 1961, 1962, and 1963. The NBSs to Lake Kyoga and Lake Albert have followed similar trends. There has been a generally declining trend since the high values of early 1960's. Since the NBSs have been derived using the historically observed records of levels and outflows, the accuracy of computation of NBSs depends upon the accuracy of the observed data. A small error in the observed water level may lead to a considerable error in the computation of NBS. For example, a 10 mm depth of water level at a surface area of 69000 km^2 represents a storage of 0.69 BCM. The equivalent discharge over a monthly time step is around 270 m^3/s.

The measurements of water levels in huge lakes such as Lake Victoria may also be affected by wind set up. Great care has to be exercised in observing the lake levels particularly during the periods of high winds. Historically, discharge was related to water levels through a rating in Owen Falls. For this reason, there is a potential for error in the computation of NBS.

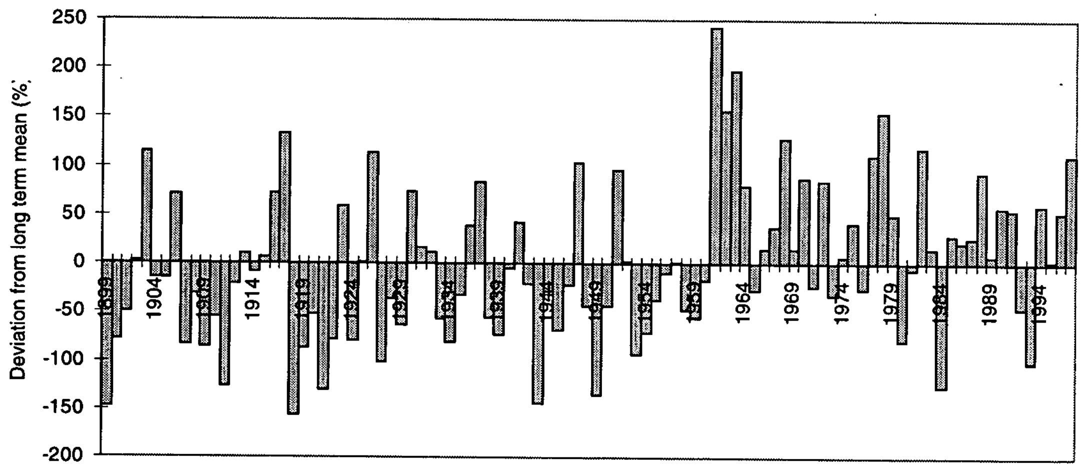


Figure 7.2 Historical variations in net basin supply to Lake Victoria

7.4 Simulation Model Runs With Historical NBS Sequence

Currently, Lake Victoria is operated according to the "Agreed Curve", which is a stage-discharge relationship that existed at the Owen Falls prior to the start of construction of the Owen Falls dam in 1954. The relationship is given by (7.11), and is illustrated by Figure 7.3.

$$Q = 630.0 + (H - 11.0) \times 403.2 \tag{7.11}$$

where Q = mean monthly discharge in m^3/s, and H is the lake level above the Jinga gauge datum. When operating to the Agreed Curve, an iterative approach is used to determine the mean monthly release and end of month water level. The model iterates to compute a mean monthly discharge corresponding with an average monthly water level determined from start and end of month levels using a tabular relationship between the gauge levels at Jinja and discharges at Owen Falls.

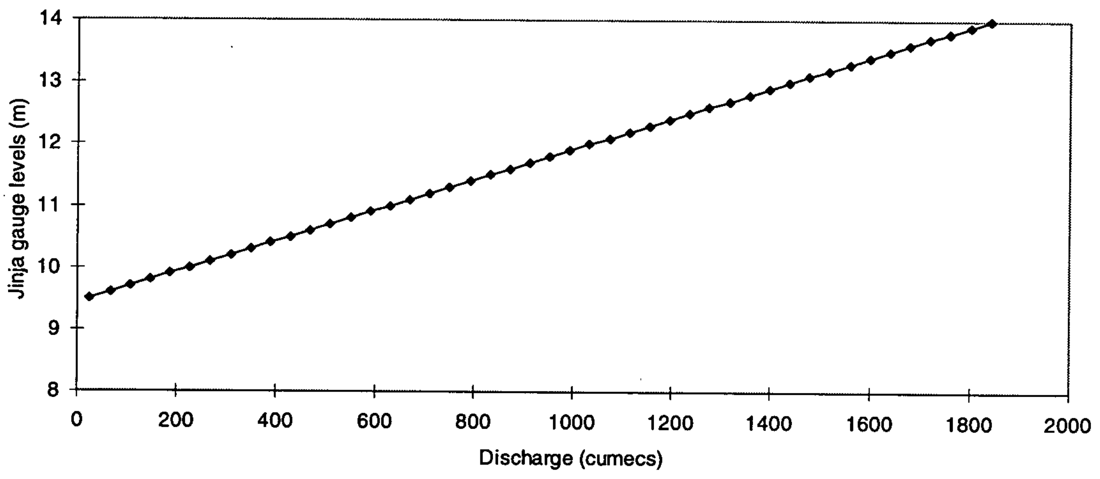


Figure 7.3 Relationship between Jinja gauge and discharge (Agreed Curve)

The water levels and discharges simulated through operation of Lake Victoria according to Agreed Curve have been compared to the historically observed values. Figure 7.4 presents simulated and observed Lake Victoria levels. There were rises in lake levels in 1906 and 1917, but the lake levels were relatively stable before 1961. The most prominent feature in the whole series is the rise of almost 2.5 m between 1961 and 1964. Since then, the levels have followed a generally declining trend except for a rise in 1979. At present, the lake levels appear to be relatively stable. Simulated and observed discharges are shown in Figure 7.5. There is a close agreement between the simulated and observed levels and discharges, particularly prior to the completion of the Owen Falls dam. The small differences in discharges after the completion of Owen Falls dam may be attributed to the minor deviations from Agreed Curve in operational practice. Thus, it may be concluded that the simulation model is reliable for the Lake Victoria.

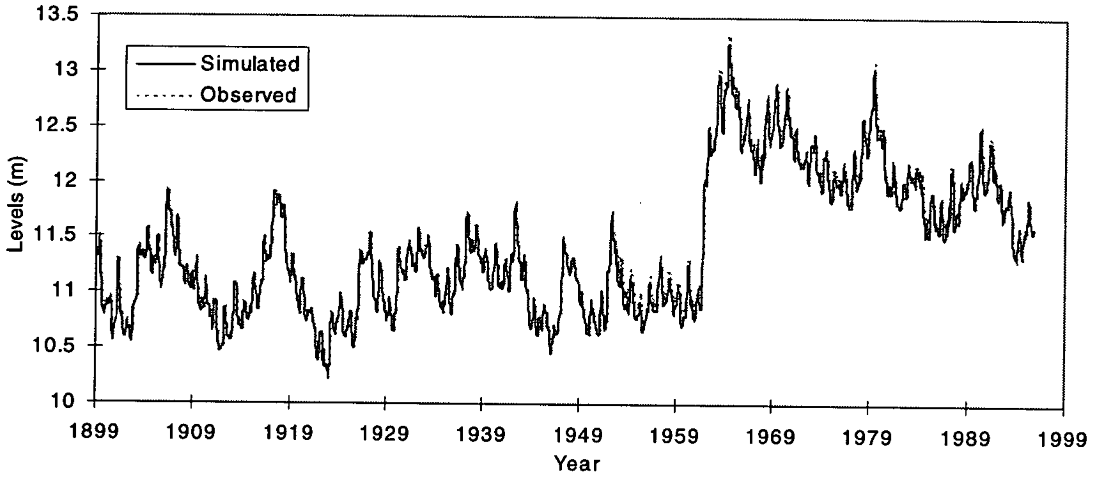


Figure 7.4 Lake Victoria simulated and observed levels

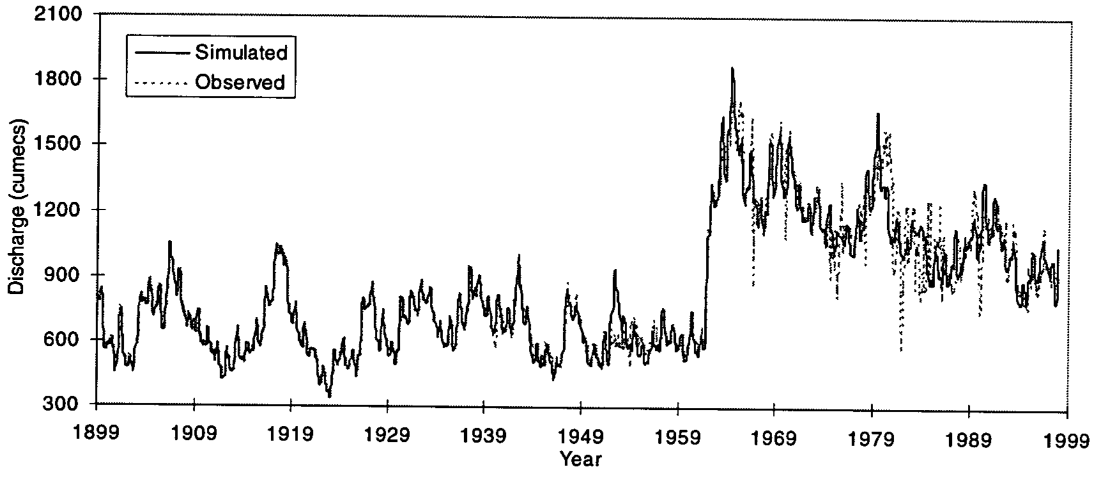


Figure 7.5 Lake Victoria simulated and observed discharges

Simulated and observed water levels for Lake Kyoga are shown in Figure 7.6, and simulated and observed discharges in Figure 7.7. It appears from Figure 7.6 that the simulated water levels lag observed water levels by about one month in the pre 1962 period, but otherwise the simulation is very good. The magnitude of the water levels is, however, same. The reason for this lag is that the missing historic water level records were infilled using the discharge data. These infilled levels are mean monthly values but are treated by the model as the end of month levels. The differences between the observed and the simulated discharges may also be attributed to the problems in observed water level and discharge records (Wardlaw 1998). Another source of problems could be the reliability of the elevation area characteristics for the lake. Given these input data problems, the performance of the model can be considered satisfactory.

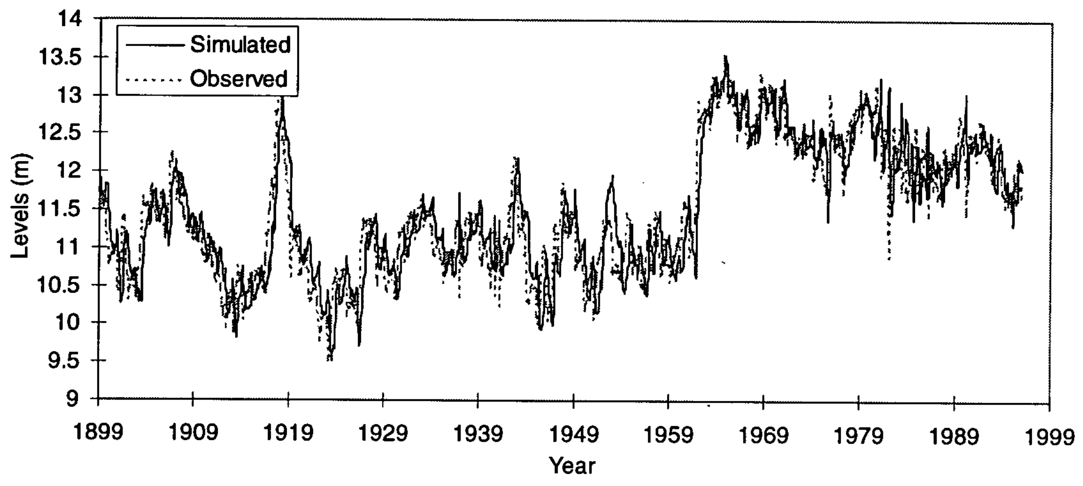


Figure 7.6 Lake Kyoga simulated and observed levels

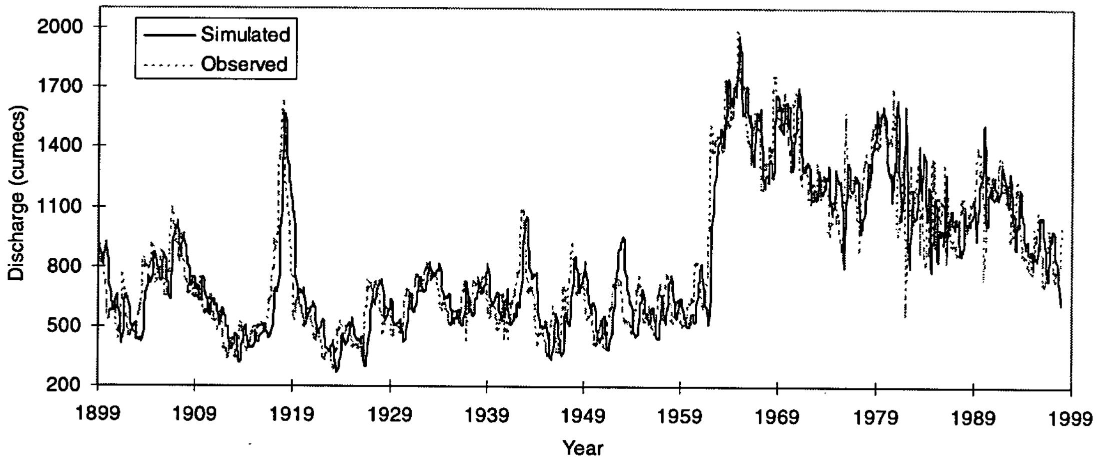


Figure 7.7 Lake Kyoga simulated and observed discharges

Lake Albert simulated and observed water levels and discharges are shown in Figure 7.8 and Figure 7.9 respectively. The problems with the data for Lake Albert are similar to those experienced for Lake Kyoga. The intended purpose of the model is to evaluate the relative impact of regulation on the lakes. For this purpose, the model can be considered to be satisfactory.

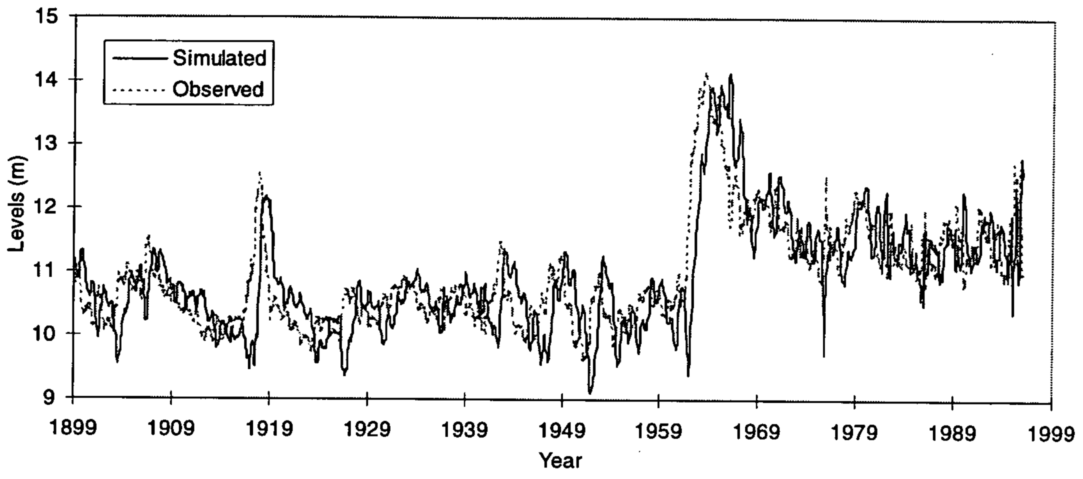


Figure 7.8 Lake Albert simulated and observed levels

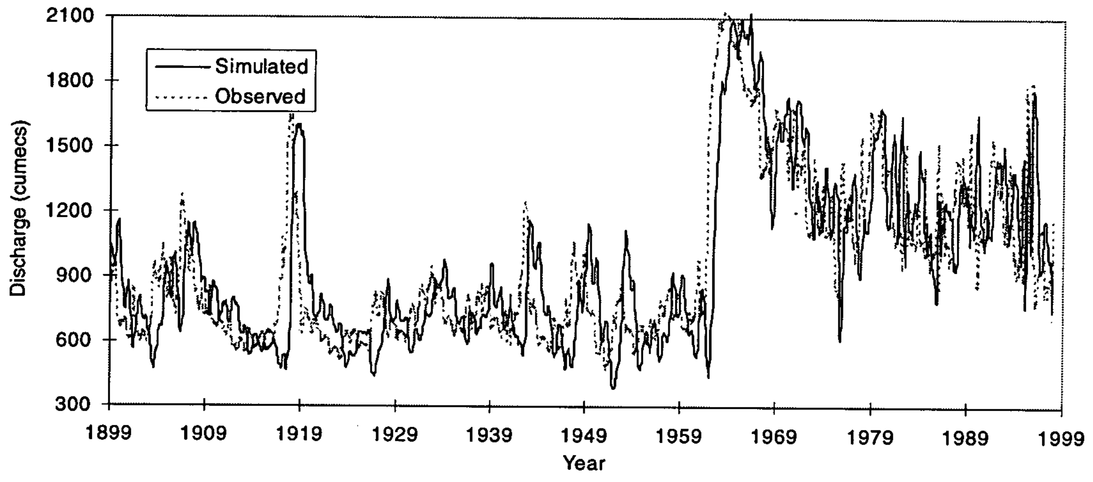


Figure 7.9 Lake Albert simulated and observed discharges

7.4.1 Evaluation of Regulation Rules

The impact of alternative regulation rules on Lake Victoria, Lake Kyoga, and Lake Albert was studied by Wardlaw (1998). A total of six regulation rules were investigated, along with operation according to the Agreed Curve. All the rules are based on different control levels for Lake Victoria. Out of these six rules, two are worthy of detailed consideration. These are described below.

(i) **Single Zone Regulation**: Lake Victoria is regulated to give a constant discharge of 660 m^3/s when the lake levels are in the range of 10.5 m to 12.0 m. Outside this range, regulation is according to the Agreed Curve. This rule keeps the levels and discharges from all lakes within the range that would occur under Agreed Curve regulation. A maximum recession rate of 0.95 is applied to discharges when the lake levels fall back into the control zone from higher levels. This avoids large fluctuations in power production particularly when the level is close to the upper control level and regulated discharge is triggered in.

(ii) **Dual zone regulation** : Lake Victoria is regulated to give a constant discharge of 660 m^3/s when the levels are in the range of 10.5 m to 12.0 m. When the levels are between 12 m and 12.9 m, a constant discharge of 1200 m^3/s is implemented. When the levels fall below 10.5 m, regulation is according to the Agreed Curve. At levels above 12.9 m, discharge is at 1800 m^3/s. This higher discharge is required so that higher extremes in water levels can be avoided, and lake levels remain within the historical range. When the levels fall back into the lower control zone from a higher control zone, a maximum recession rate of 0.95 is applied to the discharges.

The simulation model was run with single zone and dual zone regulation rules. Figure 7.10 shows the water levels and Figure 7.11 shows the simulated discharges with single zone and with Agreed Curve operation. It can be observed that there is little impact on levels but the impact on discharges in the Victoria Nile is significant. The impact of regulation on levels is minor primarily due to the huge storage capacity of Lake Victoria. The simulated water levels and discharges remain within their historically observed ranges. The discharges from Lake Victoria were routed through Lake Kyoga and Lake Albert.

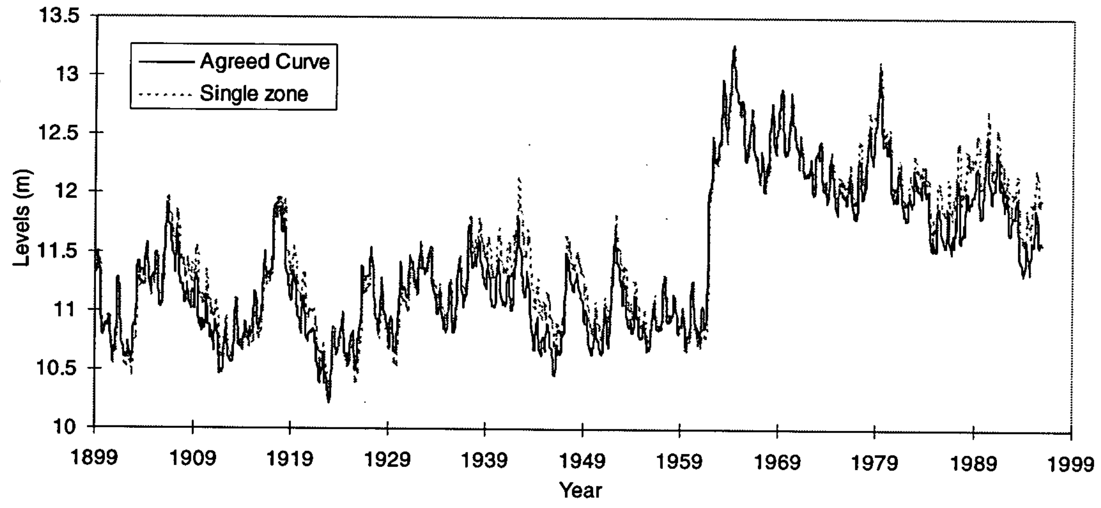


Figure 7.10 Impact of single zone regulation on Lake Victoria levels

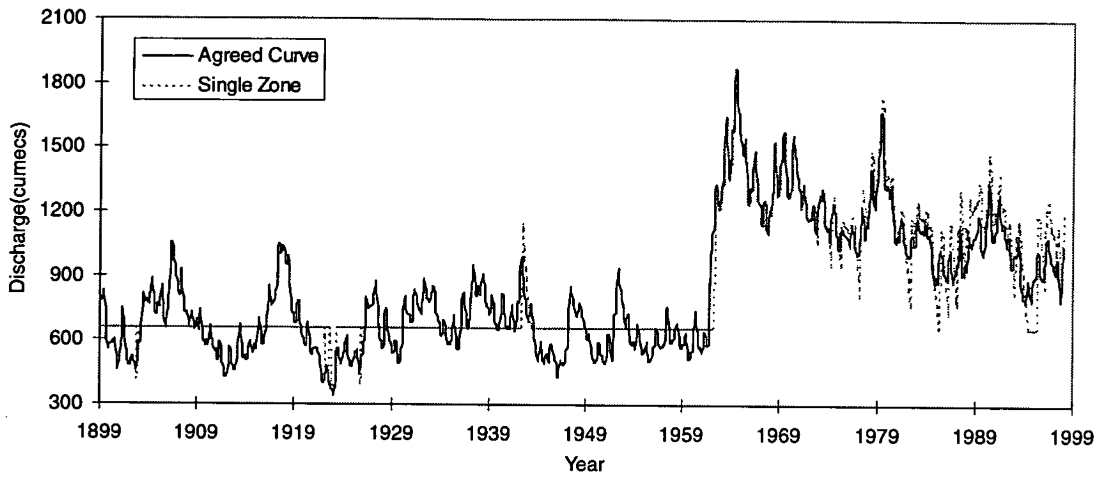


Figure 7.11 Impact of single zone regulation on Lake Victoria discharges

The impact of single zone regulation on the levels and the discharges from Lake Kyoga is shown in Figure 7.12 and Figure 7.13 respectively. Clearly, the lake levels follow observed levels very closely. The impact on discharges from Lake Kyoga is not significant, and both the levels and discharges remain within their historically observed ranges. Similarly, the influence of single zone regulation on Lake Albert is minor. The discharges and levels for Lake Albert are presented in Figure 7.14 and Figure 7.15 respectively.

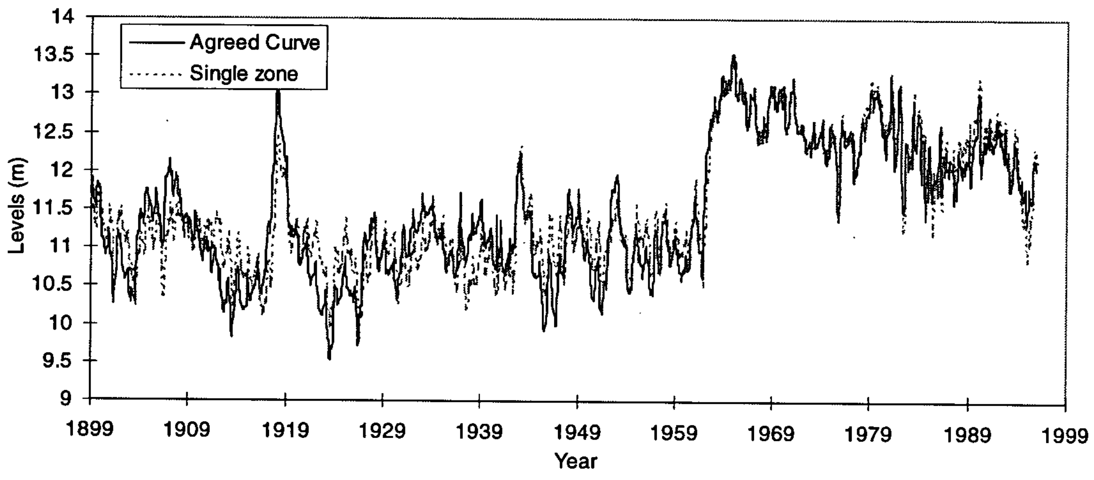


Figure 7.12 Impact of single zone regulation on Lake Kyoga levels

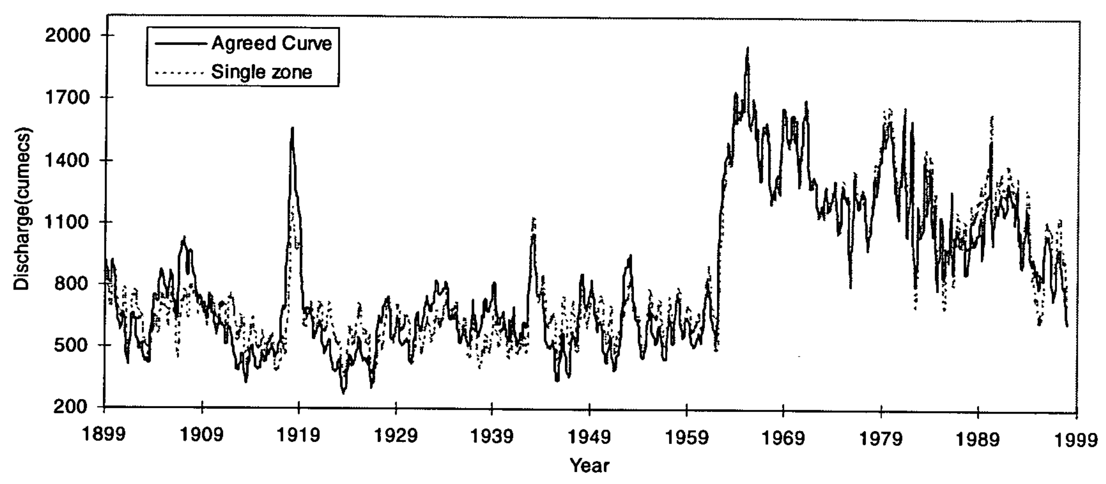


Figure 7.13 Impact of single zone regulation on Lake Kyoga discharges

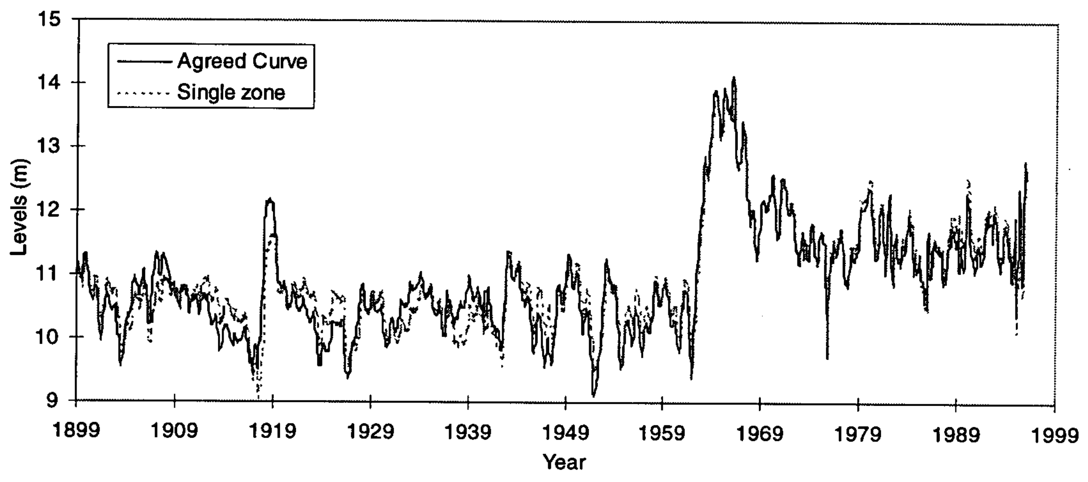


Figure 7.14 Impact of single zone regulation on Lake Albert levels

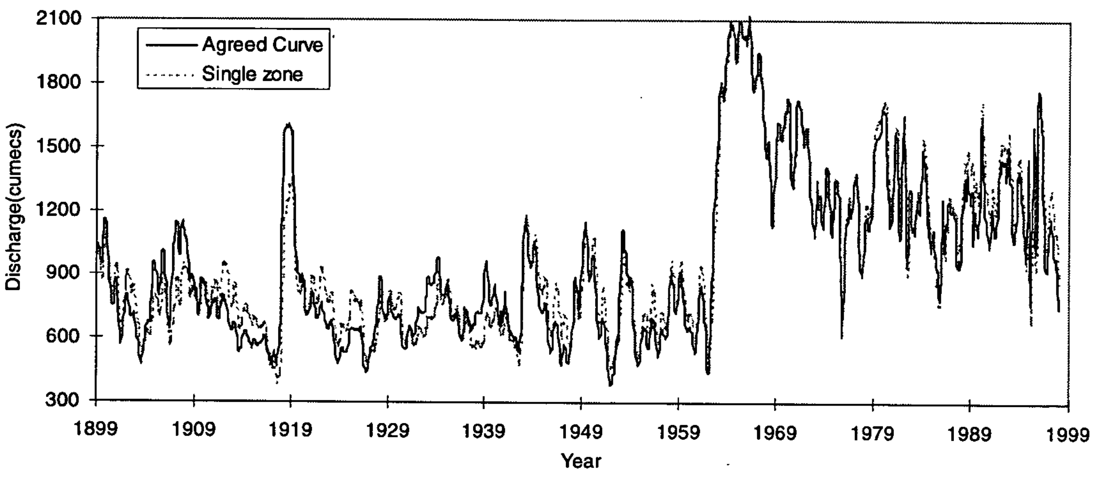


Figure 7.15 Impact of single zone regulation on Lake Albert discharges

The power production at Owen Falls dam with Agreed Curve and with single zone regulation is presented in Figure 7.16. At times when the lake levels are higher than 12.0m, power production fluctuates as the releases are determined according to the Agreed Curve. There are, however, long periods of stable power when the lake level is between 10.5 m and 12.0 m. With 95% reliability, the power produced at Owen Falls dam increased from 72.5 MW under Agreed Curve operation to 112.5 MW under single zone regulation. The increase in reliable power production at Bujagali Falls is 33.7 MW, and at Kalgala Falls it is 57.2 MW. The power stations downstream of Lake Kyoga; Ayago, Murchison and Kamdini Falls would also experience an increase in reliable power production but not to the same extent as Bujagali and Kalagala. The power duration curve for Owen Falls dam is shown in Figure 7.17, and the flow duration curve is shown in Figure 7.18.

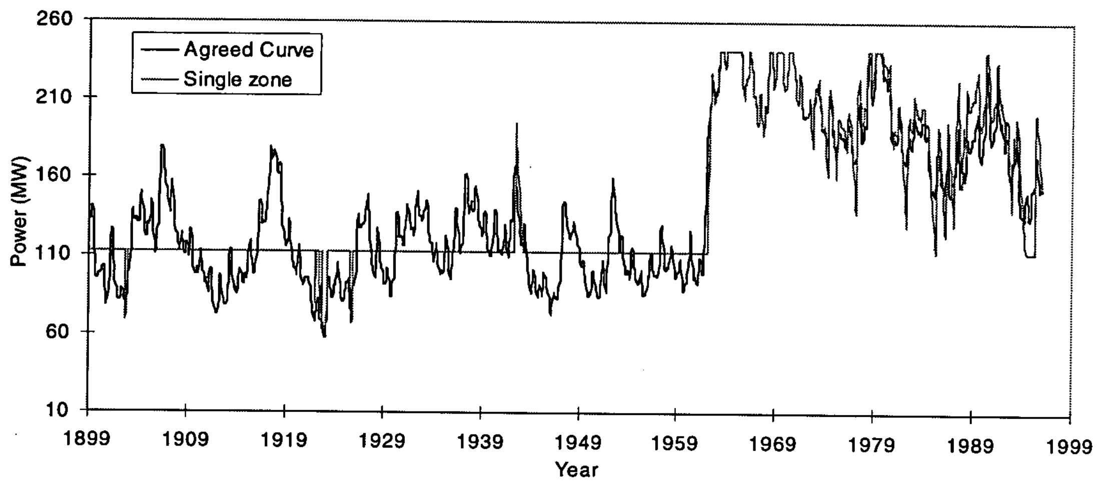


Figure 7.16 Impact of single zone regulation on Owen Falls power production

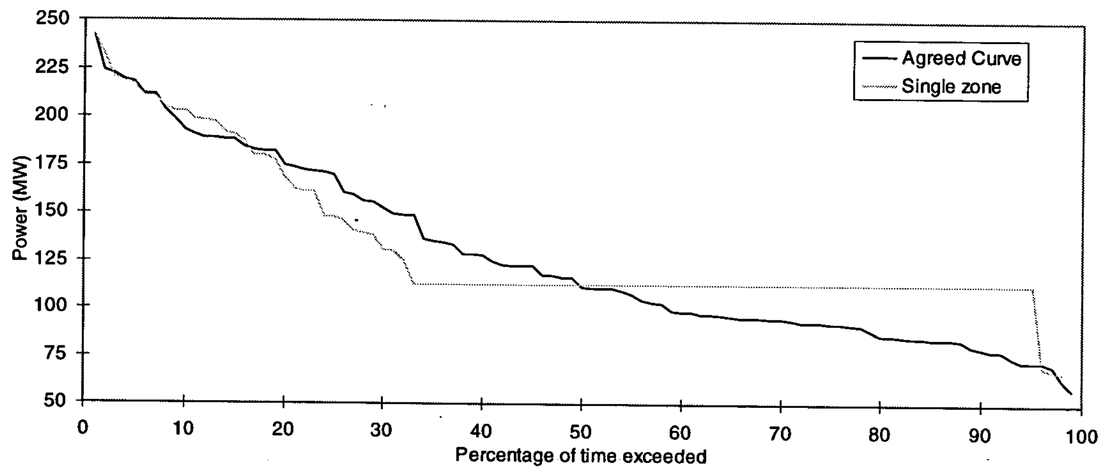


Figure 7.17 Impact of single zone regulation on Victoria power duration characteristics

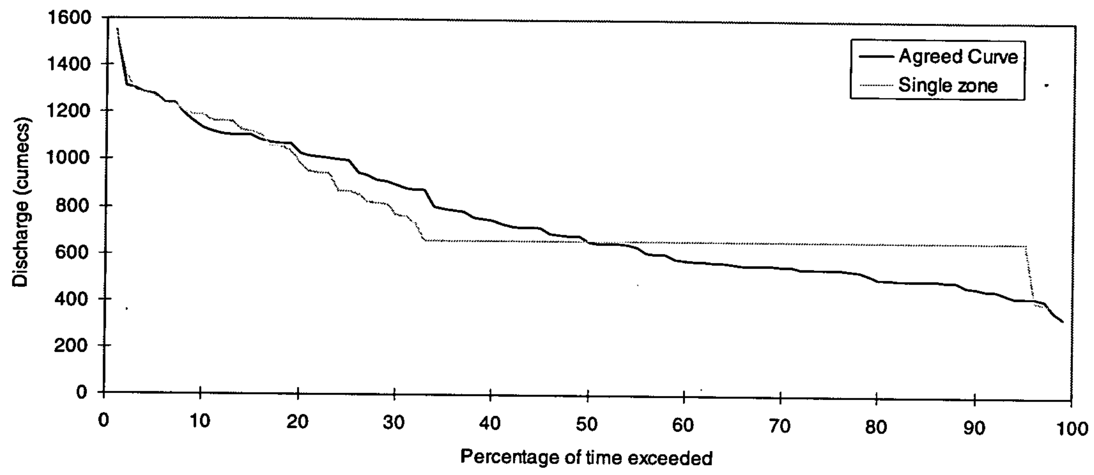


Figure 7.18 Impact of single zone regulation on Lake Victoria flow duration characteristics

With single zone regulation, fluctuating power is produced when the lake levels are higher than 12.0 m (Figure 7.16). To permit short term forecasts of reliable power to be made, flows at higher levels should also be regulated. This can be achieved by dual zone regulation wherein the lake is regulated to give a constant discharge of 1200 m^3/s for lake levels between 12.0 m and 12.9 m. This would allow use of higher flows and would result in maintaining target discharges for long periods. The impact of dual zone regulation on the levels at Lake Victoria is shown in Figure 7.19. The impact of regulation on discharges from Lake Victoria is shown in Figure 7.20. The results of dual zone regulation are very much similar to the single zone regulation with little impact on levels and significant impact on discharges.

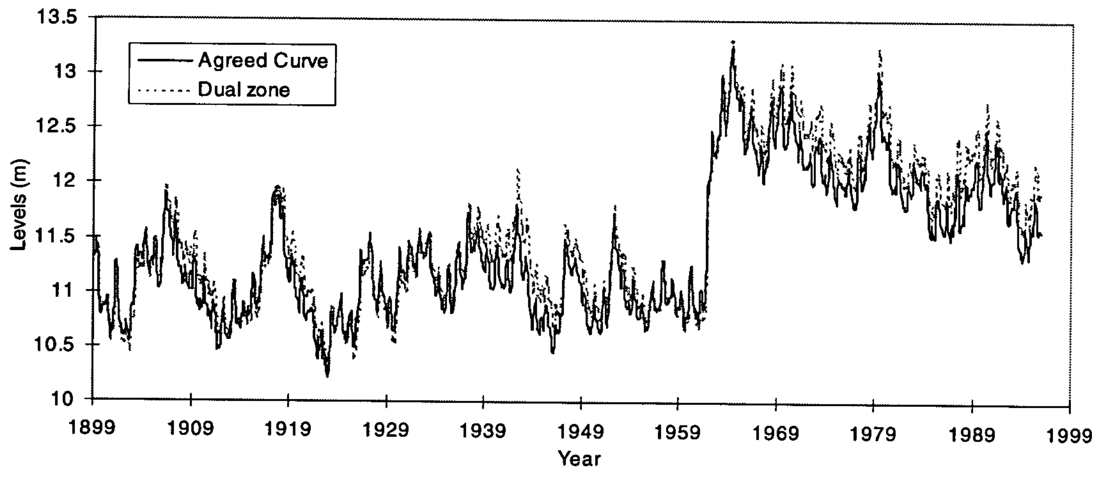


Figure 7.19 Impact of dual zone regulation on Lake Victoria levels

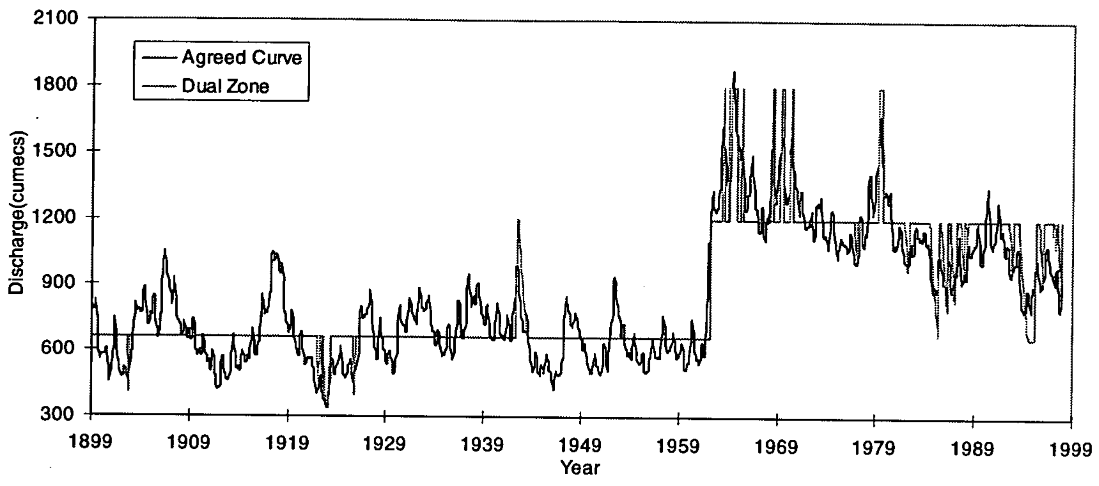


Figure 7.20 Impact of dual zone regulation on Lake Victoria discharges

The dual zone regulation stabilises the power production at higher levels too as shown in Figure 7.21. There are long periods of high power production during periods of high lake levels. The impact of dual zone regulation on power and flow duration characteristics is shown in Figure 7.22 and Figure 7.23 respectively. As with the single zone regulation, the reliable power production is increased from 72.5 MW to 112.5 MW. Also, there is a marginal increase in the average power production over that produced with the Agreed Curve. Table 7.2 gives the summary of power production at Owen Falls dam and at other power stations downstream.

Table 7.2 Summary of power production under different regulation schemes

	Potential Power Production (MW)					
Installations	**Agreed Curve**		**Single Zone**		**Dual Zone**	
	95%	**Average**	**95%**	**Average**	**95%**	**Average**
Owen Falls	72.5	141.9	112.5	143.9	112.5	143.8
Bujagali Falls	64.3	119.8	98.0	121.2	98.0	122.2
Kalagala Falls	106.3	201.6	163.2	204.3	163.1	205.1
Kamdini Falls	80.0	126.4	89.9	129.3	90.0	129.2
Ayago Falls	164.2	222.8	184.8	225.9	184.8	225.9
Murchison Falls	275.0	313.3	308.4	314.5	308.4	314.5

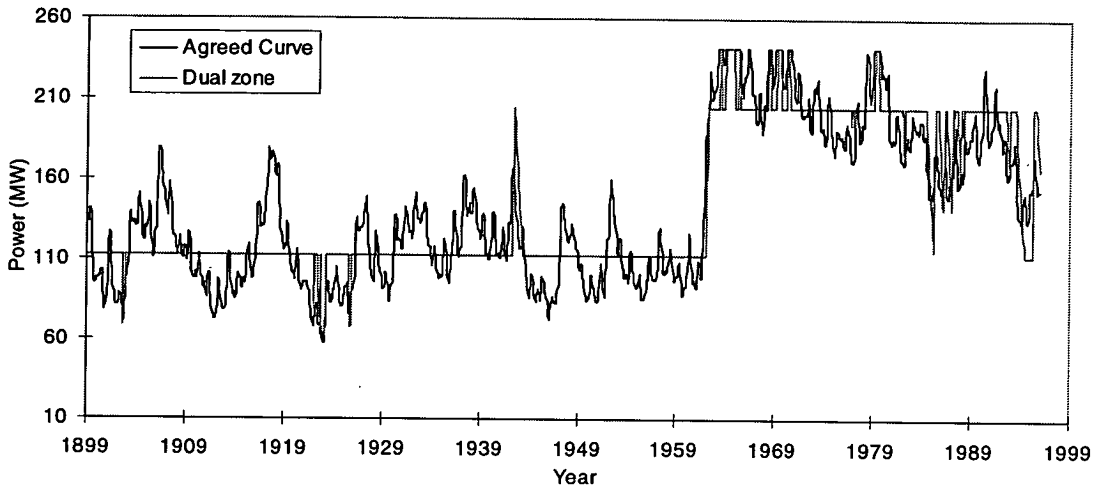


Figure 7.21 Impact of dual zone regulation on Owen Falls power production

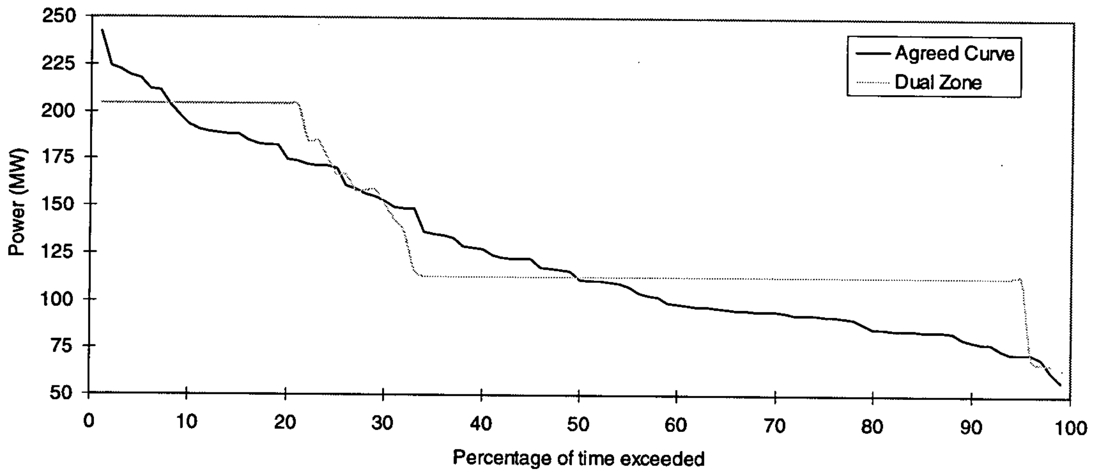


Figure 7.22 Impact of dual zone regulation on Lake Victoria power duration characteristics

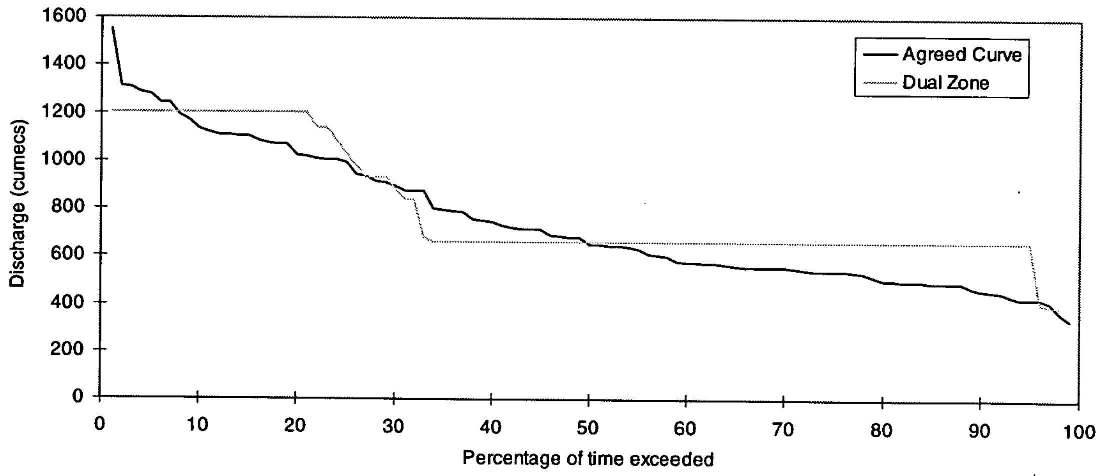


Figure 7.23 Impact of dual zone regulation on Lake Victoria flow duration characteristics

The implementation of single zone regulation is fairly straightforward and does not require sophisticated operational controls. The dual zone regulation offers the potential for higher levels of reliable power to be produced for specified periods of time. However, there is scope for improving the reliability with which power production at higher levels could be forecast. Also, it should be appreciated that the levels used for upper zones in dual zone regulation are conditioned by the period of record available. In practice, more flexibility is required and a dynamic approach to operation incorporating stochastic inflows could prove valuable. The objective of the dynamic approach would be to forecast potential power production that could be guaranteed for specified durations of time, say 3 months, 6 months, 12 months, and 24 months. These forecasts would permit planning of maintenance and outage of thermal plants. Moreover, it would be possible to sell surplus hydropower internationally as a substitute of thermal power, and to capitalise on the higher flows whenever they do occur. An approach similar to one outlined here is discussed in subsequent sections of this chapter.

7.5 Flow Forecasting Models

Streamflow forecasting techniques are an integral part of real time operation models. Stochastic streamflow models are often used in simulation studies to evaluate the response of the water resource systems to possible future scenarios. The development and implementation of such models require the historical record of the flows. Based upon the historical record, the parameters of the rainfall or streamflow forecasting model are estimated. The verification of the model is then carried out by comparing the statistics of the generated sequences with the historical sequences. Two different stochastic forecasting models are described in the following sections.

7.5.1 The Thomas-Fiering Model

The Thomas-Fiering model (Thomas and Fiering 1962; Fiering 1967) is a lag-one Markov model. In most streamflow generation techniques it is sufficient to assume that a first order Markov structure exits. The Thomas'Fiering model is fitted to the standardised monthly flows, $y_{i,j}$ given by

$$y_{i,j} = (x_{i,j} - \bar{x}_j) / \sigma_j \qquad (7.12)$$

where $x_{i,j}$ is the original flow for year *i* and month *j*, and $\bar{x}_j$ and σ_j are sample estimates of the mean and standard deviation respectively, for month *j*.

The model is based on a lag-one auto-regressive Markov process, and the standardised generated flow is given by

$$y_{i,j} = \beta_j y_{i-1,j-1} + \varepsilon_{i,j} \tag{7.13}$$

where ε_t is the random component as described by

$$\varepsilon_{i,j} = \sqrt{1-\beta_j^{\,2}} \times Z_{i,j} \tag{7.14}$$

and $Z_{i,j}$ is the randomly generated normal variate with zero mean and unit variance, β_j is the lag-one serial correlation coefficient between month *j* and *j*-1 given by

$$\beta_j = \frac{\sum d_x d_y}{\sqrt{\sum d_x^{\,2} \times \sum d_y^{\,2}}} \tag{7.15}$$

where d_x and d_y are the deviations of inflows in month *j* and *j*-1 from their respective means.

A suitable starting value and the sample estimates of monthly parameters $\overline{x_j}$ and σ_j are required to generate a continuous sequence of *T* years of synthesised monthly flows, $x_{i,j}$, where *i* = *1, T* and *j* = *1, 12* for monthly flows. Since the next period's inflow can be estimated from the previous period's inflow, (7.13) can be used recursively to generate a synthetic sequence of any length. The mean and standard deviation of the generated sequences are generally well preserved when the individual monthly flows are normally distributed. When the probability distribution of the monthly flows are skewed, they can often be normalised by transforming to the logarithmic scale. Both the models are used to first generate standardised data which is followed by inverse standardisation. If the log transform is used, inverse log transformation is carried out to obtain the synthetic sequence.

7.5.2 The ARIMA (1,1) Model

ARIMA (1,1) is an abbreviation for auto-regressive-integrated-moving-average; the values in the parentheses denote the number of autoregressive and moving average terms used in the model respectively (O' Connell 1971, 1974). Like the Thomas-Fiering model, the ARIMA model is also usually fitted to the standardised flows given by (7.12). ARIMA models combine the direct serial correlation properties of a data series with the smoothing effects of an updated running mean through the series. The generated standardised flow, $y_{i,j}$, is given by

$$y_{i,j} = \phi \; y_{i-1,j-1} + \varepsilon_{i-1,j-1} - \theta \times \varepsilon_{i-1,j} \qquad (7.16)$$

The coefficient ϕ is obtained by minimising the sum of squares of

$$r_k - r_1 \; \phi \; (k-1)$$

from $k = 2$ to $k = L_{max}$ where r_k is the lag k autocorrelation function and L_{max} is the maximum lag. In program SYNHY1 which has been used to generate flows, L_{max} has been arbitrary set equal to 100. The second coefficient θ is given by

$$\theta = (a - \sqrt{a^2 - 4b^2}) / 2b$$

where $\mathrm{a} = 1 - 2\phi \; \mathrm{r}_1 + \phi^2$

and $b = \phi - r_1$

The coefficients ϕ and θ used in an ARIMA model do not vary from month to month. So, the model cannot be expected to preserve the statistical parameters of the historic data. The model, however, provides a better means of modelling the Hurst Effect (Hurst 1951), which is described next.

Hurst (1951) while studying the long-term storage requirements on the Nile River found that

$$R = \sigma \left(\frac{n}{2}\right)^h$$

where R is the range of cumulative deficits from the mean flow, σ is the standard deviation, and n the length of the series. The ratio $\frac{R}{\sigma}$ is known as rescaled range, and exponent h is known as the Hurst coefficient. The value of h for a statistically independent process such as Markov process is 0.5. For some 800 time series analysed by Hurst, the average value of h was around 0.73. The discrepancy between the theoretical results and the values observed by Hurst has become known as the *Hurst phenomenon.* This discrepancy could be due to the non-stationarity of the series or due to the statistical dependence of the series (Klemes 1974). A Hurst coefficient between 0.5 and 1.0 corresponds to a long term persistence in series, which is also known as Joseph effect (Mandelbrot and Wallis 1968).

The Hurst phenomenon is, however, questionable . Since its discovery, the puzzle of Hurst phenomenon has haunted hydrologists and mathematicians as the physical significance of h is not completely known although it gives some measure of long term persistence in the natural time series. The coefficient can be perceived as an index to drought phenomenon, and models that can replicate h have the capability of replicating historically observed droughts. The sequences generated by the Thomas-Fiering model exhibits values of h of around 0.5. On the other hand, ARIMA models tend to preserve persistence in the series and produce h values close to those observed by Hurst.

The Hurst phenomenon occurs due to the persistence in streamflows caused by storage effects. The Lake Victoria models considered here are, however, driven by the NBS which is to a large extent based upon the rainfall, particularly for Lake Victoria. As shown in Figure 7.2, there is large variability in the NBS which clearly indicates the absence of persistence in NBS and the rainfall. The Hurst phenomenon cannot be attributed to one specific physical cause but one of the several causes reported for its existence is the characteristics of a particular storage system (Klemes 1974). For Lake Victoria system, the storage upstream of the Lake is insignificant; only at the point of supply is the storage significant. The Hurst phenomenon therefore does not appear to exist for NBS to Lake Victoria.

The standardised monthly data is generally used to generate synthetic streamflow sequences with forecasting models such as Thomas-Fiering or the ARIMA models. Following standardisation, the data has zero mean and unit standard deviation. A correlogram analysis may be used to obtain an insight into the character of the data. Figure 7.24 shows the correlogram of the observed historic data, standardised historic data, and the synthetic data generated by the Thomas-Fiering model for lags of upto 100.

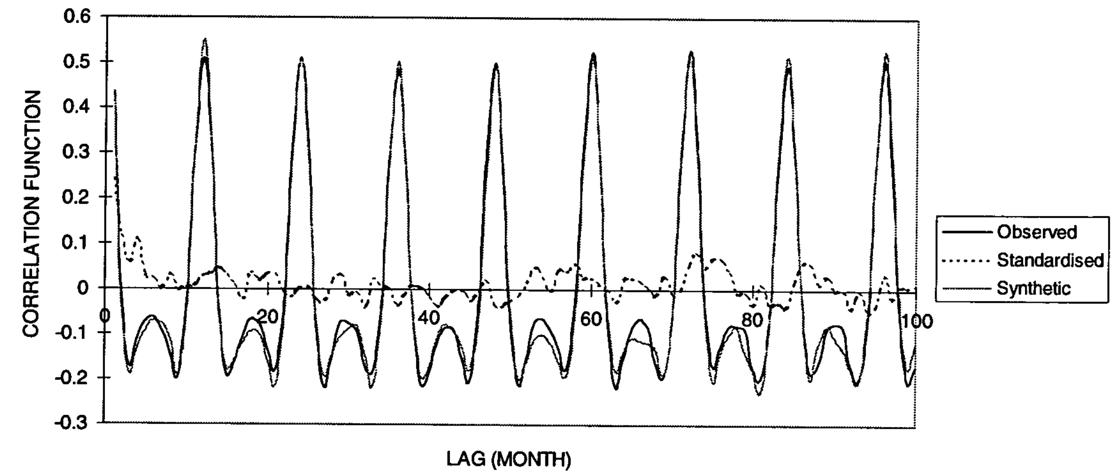


Figure 7.24 Correlogram of observed, standardised, and synthetic data

A key assumption in the correlogram analysis is that the standardised data is stationary. The correlogram for the historic data shown in Figure 7.24 has a periodicity of 12. The periodicity is absent in the standardised data and its correlogram shows a decay with lag similar to that which would be expected for a stationary Markov process. The assumption of a stationary standardised monthly data is therefore justified.

7.5.3 Determination of Appropriate Forecasting Model

The FORTRAN programs SYNHY1 and SYNHY2 (Mott MacDonald 1988) have been used to generate forecasts of NBS to Lake Victoria with a view to determining the most appropriate forecasting model for the lake. Program SYNHY1 is used to set up the parameters of the Thomas-Fiering and the ARIMA models. Using these parameters, program SYNHY2 can be used to generate several synthetic sequences. Ten NBS sequences starting with different initial random number seeds were generated using the Thomas-Fiering and the ARIMA(1,1) models. The generated sequences were then compared to the historical sequence in terms of basic statistics such as mean and standard deviation. The statistics of these sequences are presented in Table 7.3. The monthly mean and standard deviation of the historical sequence are 2245 million cubic metre (MCM) and 6581 MCM respectively.

Table 7.3 Statistics of synthetic sequences

SEQUENCE	**Thoamas-Fiering**			**ARIMA (1, 1)**		
NUMBER	**Mean (MCM)**	**Std. Dev. (MCM)**	**Skewness Coeff.**	**Mean (MCM)**	**Std. Dev. (MCM)**	**Skewness Coeff.**
1	2139	6548	0.659	2109	7395	0.513
2	1896	6427	0.753	1635	7279	0.711
3	2036	6640	0.698	1951	7280	0.635
4	2152	6647	0.659	2132	7441	0.568
5	2107	6411	0.587	1955	7194	0.505
6	1918	6429	0.595	1728	7242	0.473
7	2153	6480	0.621	2080	7233	0.537
8	1992	6673	0.685	1921	7549	0.641
9	2149	6787	0.783	2122	7603	0.749
10	2243	6555	0.650	2282	7282	0.583
Historical	2245	6581	0.98	-	-	

The statistics presented in Table 7.3 clearly indicate that the mean and standard deviation of the sequences generated by the Thomas-Fiering model were in good agreement with the corresponding historical values. However, the standard deviations for the sequences generated by the ARIMA model showed significant differences from the historical values (Table 7.3). This is due to the reason that the ARIMA (1,1) model tends to generate long periods of extreme values (very low or very high) of NBS. Also, the ARIMA (1,1) model, by its nature, cannot be expected to preserve the historical means of NBS for different months. On the other hand, the Thomas-Fiering model preserves the historical means well (Table 7.3). For the present study, it has been considered appropriate to use the Thomas Fiering model to generate sequences of NBS in all cases to follow.

7.6 Simulation Model Runs With Synthetic NBS Sequences

In this section, single zone regulation is applied with synthetically generated NBS sequences. The basic idea behind using these synthetically generated sequences is that each of these sequences may be considered as representative of a likely future scenario. Using a single historical streamflow record to derive reservoir operating rules is generally not considered to be satisfactory, and it is common practice to evaluate operation with synthetically generated alternative flow sequences.

Ten sequences of 99 years NBS each starting with a different number seed were generated using the Thomas-Fiering model. These sequences were combined to obtain a sequence of 990 years of monthly NBS data. Using this sequence, the operation of Lake Victoria was carried out according to single zone regulation. The output obtained from the 990 year simulation by using synthetic inputs was analysed to determine the power that could be produced from the Owen Falls under single zone regulation at different levels of reliability. The power production at 90% and 95% exceedence levels along with the long term annual average with the synthetic sequences and the historic sequence is presented in Table 7.4. The power duration curve is shown in Figure 7.25.

Table 7.4 Summary of power produced in MW with 990 year synthetic sequence

Synthetic sequence			Historical Sequence		
Average	**90%**	**95%**	**Average**	**90%**	**95%**
160.5	112.5	112.5	141.9	79.5	72.5

As shown in Table 7.4, the power produced with 90% and 95% reliability using the 990 year synthetic NBS sequence is 112.5 MW. The power production with the synthetic sequence is considerably higher than with the historical NBS sequence. From Figure 7.2 it can be seen that there is a steep rise in NBS between 1961-1963 mainly due to a period of heavy rainfall during these years. In 1961, there was a deviation in NBS of around 250% from the long term mean. Statistically, this affects the mean and standard deviation of the sample used to generate the forecasts of NBS with the Thomas-Fiering model. It appears that the synthetic sequence does not contain extremely low values of NBS as had been experienced historically. This is the likely reason why the average and the reliable power production with the synthetic sequence are considerably higher than with the historical sequence.

The affect of synthetic sequences on the levels in Lake Victoria was also investigated. The historically observed minimum levels for Lake Victoria is 10.22 m. The level duration curve obtained by using the 990 year synthetic sequence is shown in Figure 7.26. On the basis of synthetic sequence generated by the Thomas-Fiering model considered here, the probability that the lake level will go down below the historical minimum level under single zone operation is less than 1 percent (Figure 7.26). This is interesting because the stochastic flow generation models such as the Thomas-Fiering model have the ability to produce events more extreme than those actually experienced historically. However, it appears that the generation of NBS with the Thomas-Fiering model is affected by the high values experienced during 1961-1964, and therefore extremely low values of NBS are not generated.

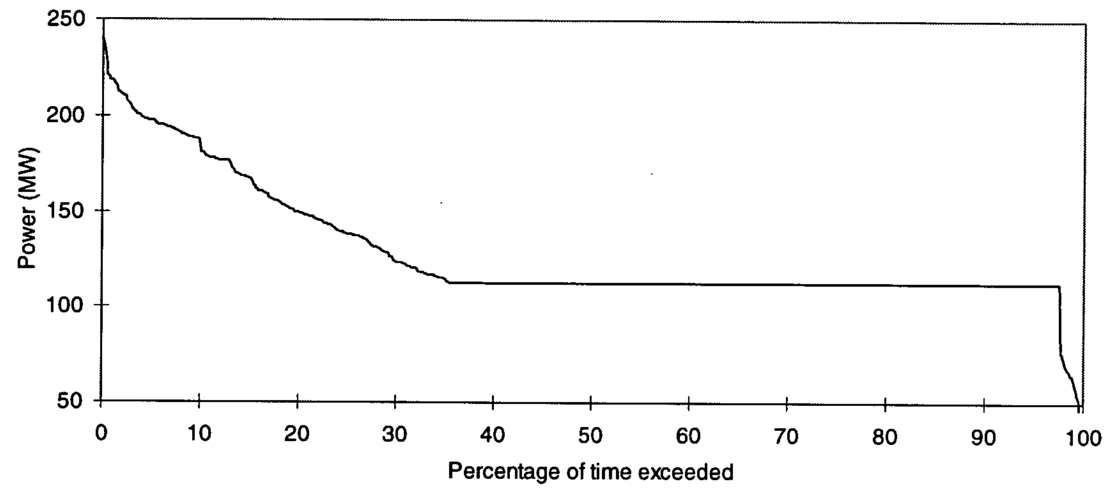


Figure 7.25 Power duration curve, simulation with 990 year synthetic sequence

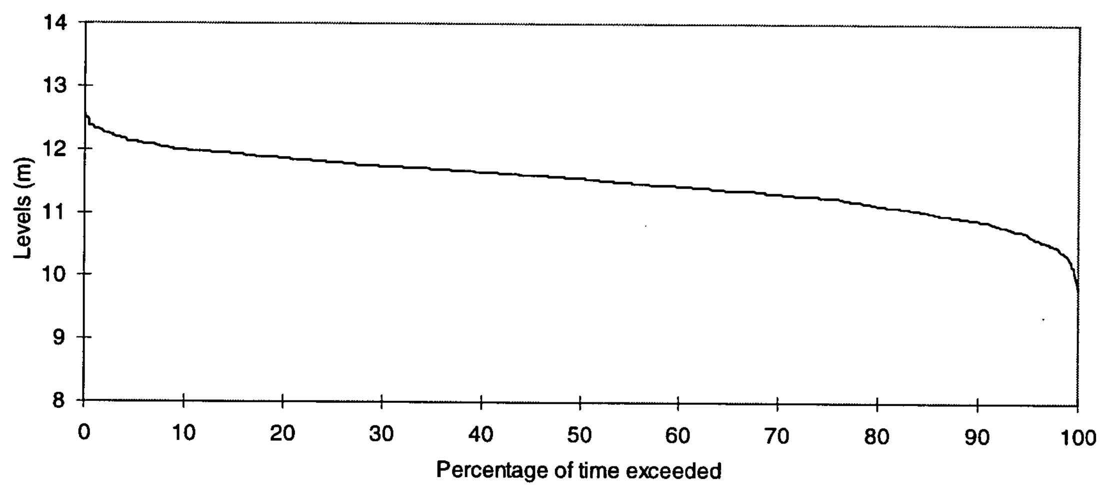


Figure 7.26 Level duration curve, simulation with 990 year synthetic sequence

7.7 Real Time Operation

In real time reservoir operation, stochastic models are used to obtain scenarios of possible future inflows on the basis of which effective operating decisions can be made. Real time operating models can help in the management of extreme events, such as floods and droughts that occur over relatively short time periods. They can also be used to manage complex systems on a continuing basis. In real time operation of reservoir systems, the models take into account future forecasts of inflows to determine optimal operating policy over a specified length of time. While only the decisions in the current period are implemented, the impact of these decisions in the future is considered. Real time models are run sequentially and the information concerning the current state of the system, current and future inflows are updated each time the model is run. A flow chart outlining the real time operation procedure is shown in Figure 7.27.

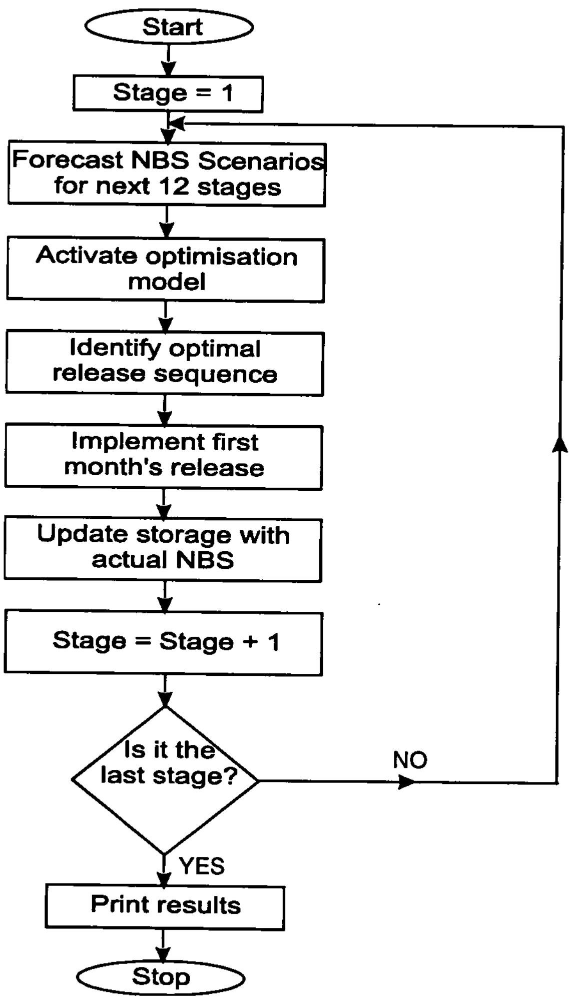

Start
Stage = 1
Forecast NBS Scenarios for next 12 stages
Activate optimisation model
Identify optimal release sequence
Implement first month's release
Update storage with actual NBS
Stage = Stage + 1
Is it the last stage?
NO
YES
Print results
Stop

Figure 7.27 Real time operation procedure

Since the forecasts of inflows are based on the actually observed values in the previous time step, the occurrence of any extreme phenomenon can be implicitly taken into account. For example, a very low inflow in the previous time step is likely to generate future forecasts of inflow having low magnitudes.

7.7.1 The System Dynamics

The dynamics of the Equatorial Lakes system can be described by the following mass balance equation.

$$S_i(t+1) = S_i(t) + I_i(t) + MR_i(t) - E_i(t) \tag{7.17}$$

where the terms have their usual meanings and **M** is a n order square matrix describing the system configuration with -1 along the diagonals +1 in the position ith column and jth row if the release from reservoir i goes into reservoir j. The rest of the martrix elements are zero. From this consideration, the matrix M for this system could be expressed as follows.

$$\mathrm{M}=\begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

The constraints on storages and releases can be described in the usual manner as follows.

$$S_{i,\min} \le S_{it} \le S_{i,\max} \tag{7.18}$$

$$R_{i,\min} \le R_{it} \le R_{i,\max} \tag{7.19}$$

where $S_{i,min}$ and $S_{i,\,max}$ represent the lower and upper bounds on storages; $R_{i,min}$ and $R_{i,\,max}$ are the lower and upper bounds on releases from lake i.

The operation of Equatorial Lakes system may be described by (7.17), (7.18), and (7.19). Most models that simulate the process of real time operation use an optimisation model for determining optimal decision in each time period. For this research, the DDDP and the GA models have been used to determine optimal real time operating decisions. Two case studies are presented. In the first case, the real time operation of Lake Victoria is considered. The combined operation of Lakes Victoria, Kyoga and Albert is considered in the second case.

7.7.2 The Single Lake System

In this case study, real time operation of Lake Victoria is considered. The objective function is to maximise the hydropower generation from the existing hydropower facility at the exit of Nile and from the proposed extension at the Owen Falls. Mathematically, the objective function could be expressed as

$$Maximise\ J_1 = p_i(t) \tag{7.20}$$

where $p_i(t)$ is the power produced at lake i in time step t.

The energy generation at Owen Falls dam is a linear function of release as given as (7.7), and is effectively independent of lake elevation. Currently, there are no operational controls at Lake Kyoga and Lake Albert. It was considered reasonable to use the optimal releases made from Lake Victoria as an input to Lake Kyoga. Deterministic simulation is then carried out for Lake Kyoga and Lake Albert.

To prevent the lake levels from falling below the minimum allowable levels, a penalty function approach was used. This requires modification of the objective function given by (7.20). A penalty term was added to the objective function which penalizes releases that exceed those determined by the Agreed Curve whenever the lake levels fall below 10.5 m. Since the objective of optimisation is to maximise the hydropower production, the model tends to release the maximum possible amount of water whenever it is feasible to do so. For example, at lake levels above 11.0 m it is possible to generate full power without violating the lower bound on the lake levels. At levels lower than 11.0 m, the model still tends to produce maximum amount of power until the lake levels go down the historical minimum value. The reason for these high release even at low lake levels is that the power function is independent of lake elevation. It is therefore important to derive an appropriate penalty function which gives the maximum value of the objective function when the release generated by the optimization model coincides with that determined by the Agreed Curve. The modified objective function may be expressed as:

$$Maximise\ J = \sum_{i=1}^{n} \sum_{t=1}^{N} (J_1 - J_2) \tag{7.21}$$

where n is the number of lakes in the system, N is the length of operating horizon, and

$$J_2 = \begin{Bmatrix} k(r - r_{allow}) & if \quad r > r_{allow} \\ p(r_{allow} - r) & if \quad r < r_{allow} \\ 0 & if \quad r = r_{allow} \end{Bmatrix} \tag{7.22}$$

where r_{allow} is the release allowed by the Agreed Curve for the given lake level, k and p are penalty multipliers which determines the amount of penalty to be imposed when the release generated by the optimization model exceed those allowed by the Agreed Curve. The system objectives like the requirements that the lake levels and releases should remain within their historically observed ranges are incorporated into the model through the constraints described by (7.18) and (7.19).

7.7.2.1 The DP Approach

The DP algorithm is applied to the operation of Lake Victoria using a 12 month planning horizon. To apply DP to the problem formulated above, the storage of Lake Victoria must be discretized into a finite number of states. The elevation data for the lake is available in increments of 0.1 m. For each lake elevation, the corresponding value of the state variable is also available. By interpolating between the given data, the storage and elevations of Lake Victoria were discretized into 2000 levels each. This gives an incremental storage of 0.17 BCM which is equivalent to a release of 65.6 m^3/s over a monthly time step. The levels are discretized between 9.113 m and 14.113 m in increments of 2.5 mm. It is necessary to use a large number of discretizations because Lake Victoria has huge storage capacity (3121 BCM) and a coarser discretization than the one adopted here might lead to considerably inaccurate results. The computed loss in storage also becomes seriously in error if the discretization is too coarse. A finer discretization than that adopted here would increase the computational burden of optimising the recursive equation.

The starting lake levels are taken as 11.4 m for Victoria, 11.74 m for Kyoga, and 11.0 m for Albert. These values have been obtained from the historic water level records for the lakes. The respective datum are 1122.887 m, 1020.61 m, and 609.82 m. The upper and lower bounds on releases and levels are set equal to the historically observed maximum and minimum values. For each time step of the operating horizon, the Thomas-Fiering model is used to generate a forecast of the sequence of NBS to Lake Victoria for the next 12 time steps. The basis of these forecasts is the 97 year (1899-1995) long monthly net basin supply record. Using the forecast sequence of next 12 months NBS, the DP model is used to determine the optimal release sequence that maximises the power production over the operating horizon. From these 12 time step optimal release sequences, only the release of first time step is actually implemented. The system state is then updated using the actual NBS for the current time step (Figure 7.27). The procedure is repeated for every time step of the control horizon (12x99=1188 time steps) and as a result optimal reservoir levels and releases for the lake are obtained for each month of the operating horizon. The optimum releases obtained from the DP model are then used as an input to Lake Kyoga. Deterministic simulation is then carried out for Lake Kyoga and Lake Albert. The total execution time for completing a DP run on a Pentium based 586 PC was around 115 minutes.

7.7.2.2 The GA Approach

The chromosome length in terms of the number of genes required to represent a solution to the problem is 12 (1 lake and 12 time steps) as only the operation of Lake Victoria is optimised. The same GA code as used for the problems described in chapters 4 and 6 has been used for this case study. Tournament selection, uniform crossover, modified uniform mutation and elitism has been used. A probability of crossover of 0.7, and a mutation probability of 0.083 which corresponds to 1 mutation per chromosome has been used. With a population size of 100, the GA model was run for 200 generations using the same objective function as that used for the DP model.

The initial lake levels and other data are the same as used for the DP model. Discretization of storage space is not required in the GA approach. The procedure for real time operation using GA is similar to the one used for the DP approach. A planning horizon of 12 months was used. The whole GA procedure is invoked at each time step of the control horizon. At the beginning of the run, the forecasts of NBS to Lake Victoria for the next 12 months is made using the Thomas-Fiering model. An optimal (or a near optimal) solution in the form of a release policy is obtained for these sequences of NBS by running the GA model for a predetermined number of generations. The optimal release of only the first time step is implemented and the state of the system is updated using the actual NBS to the system. The whole GA procedure is restarted with new initial levels, storages and a forecast NBS sequence for the next 12 months. The procedure continues until every time step of the control horizon is covered. The execution time for the GA model was around 155 minutes which is longer than for the DP model.

Although discretization of releases is not required, it was considered advantageous to make releases from the lake in increments of 50 m^3/s. This required slight modification to the initialisation function of the GA module. Instead of allocating continuous values, the genes were allocated values in increments of 50 m^3/s. This was achieved by using a random number function that returned releases in multiples of 50 cumecs. The rationale behind using discrete values of releases is that it speeds up the GA procedure. Also, it is impractical to make releases with high degree of precision. The modified mutation operator was used to change the value of the gene selected for mutation by a fixed amount of 50 m^3/s. This change could be either positive or negative depending upon the outcome of the random number generator.

7.7.2.3 Evaluation of model runs

The lake levels produced under real time control by the GA and the DP models are shown in Figure 7.28. It is clear from the figure that the levels produced by the GA and the DP model match each other closely. There is, however, some deviation in levels from their historical range, and the minimum level is lower than the historically observed minimum levels by about 0.2 m . This occurs because if the NBS is negative, the lake levels may fall below a predefined lower bound when the updating of storage with actual NBS takes place.

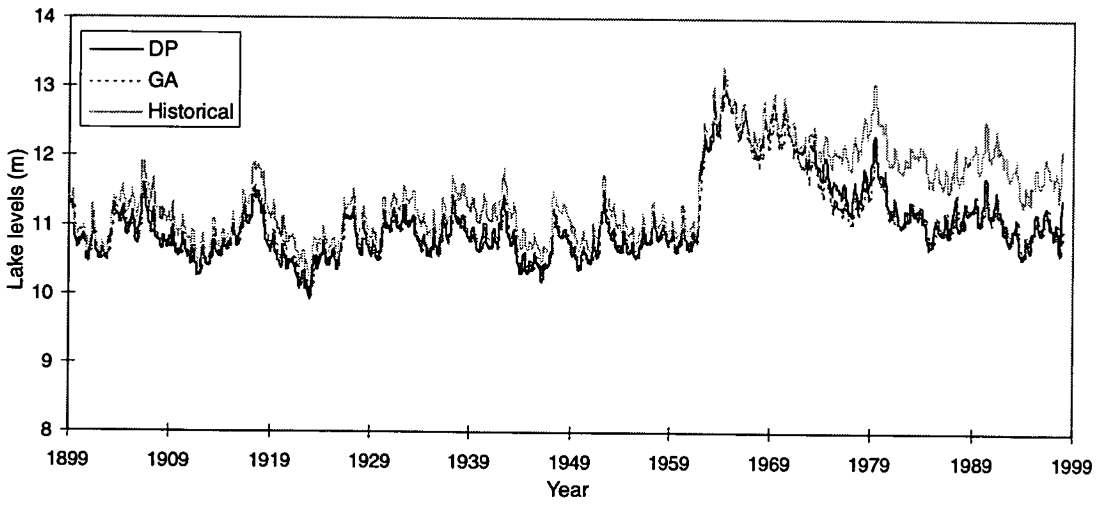


Figure 7.28 Lake Victoria levels, single lake system

Figure 7.29 shows the power sequences produced by the GA and the DP models. There is large variability in power production with either of the models. Both models tend to generate full power whenever there is enough storage to do so thereby forcing the lake to lower levels more often. This results in fluctuating power production as shown in Figure 7.29. With single zone regulation, a constant release of 660.0 m^3/s is made when the lake levels are between 10.5 m and 12.0 m. Full power is produced with single zone regulation only when the lake levels are higher than 12.96 m. On the other hand, the optimisation models (both GA and DP) tend to release maximum possible amounts even when the lake levels are lower than 11.0 m. The average power production with the DP model is 147.5 MW which shows that there is an improvement over the single zone operation. On the other hand, the energy which will be available 95% of the time is only 64.7 MW which is significantly lower than with the single zone regulation. With the GA model, the average power production is 146.2 MW, and the reliable power output is 64.1 MW. The reliability of power production is low with both the models because of the extreme variability in power generation. In practice, this type of operation would not be satisfactory due to the requirements of producing power at reliable levels.

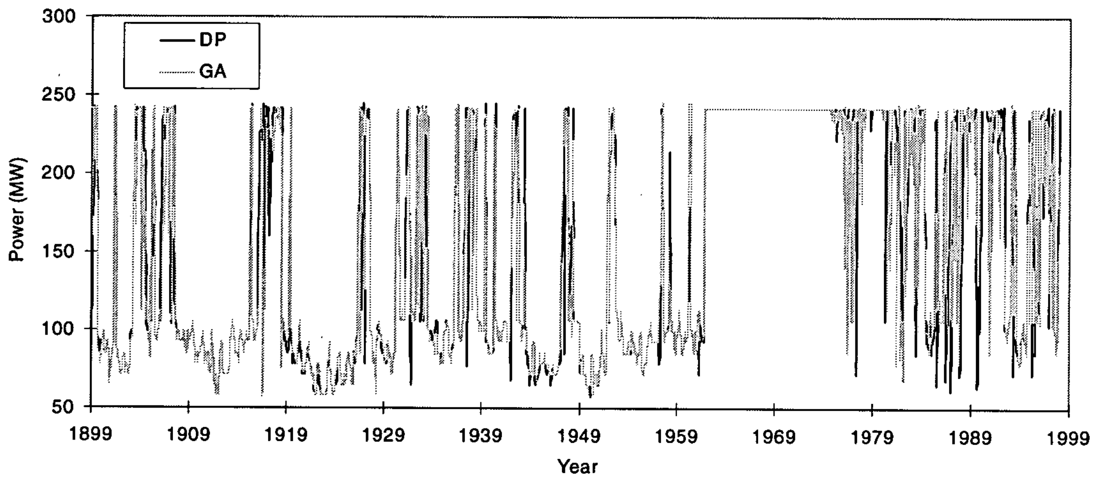


Figure 7.29 Power production at Owen Falls, single lake system

7.7.3 The Three Lake System

In this section, the combined operation of the Equatorial Lakes system, assuming that operational controls exist at the outflows from Lake Kyoga and Lake Albert, is considered. Currently, there are no controls at these lakes. In addition to maximising the power production at Owen Falls dam, the operation of Lake Kyoga and Lake Albert is considered.

The objective of operation for these lakes is to maintain the levels in the vicinity of their mean historical values. Georgakakos (1996) used the following function for achieving this aim. The same function has been used in this study.

$$J_3 = \left(\frac{H_K(t) - H_K^*}{\Delta H_K}\right)^2 + \left(\frac{H_A(t) - H_A^*}{\Delta H_A}\right)^2 \qquad (7.23)$$

where $H_k(t)$ is the level in Lake Kyoga, H_k^* is the historical mean level, and ΔH_k is the range of the historical levels. The lake levels depend upon the storage which is a function of NBS and releases. Therefore, the model maintains the lakes levels close to a given target by controlling the releases made from the Lake.

The overall objective in this case study is to maximise the power production at Owen falls dam and to maintain the levels in Kyoga and Albert as close to the historical mean levels as possible. From the historic data, the long term mean levels for Lake Kyoga and Lake Albert are 11.495 m 10.941 m respectively. Mathematically, the objective function J for the system can be expressed as

$$Maximise\ J = \sum_{i=1}^{n} \sum_{t=1}^{N} (J_1 - J_2 - J_3) \qquad (7.24)$$

subject to the constraints described by equations (7.18) and (7.19). The functions J_1 and J_2 are given by (7.20) and (7.22). In (7.24), n is 3 as all the three lakes are included in the model, and N is 12.

7.7.3.1 The DDDP Approach

DP cannot be used to optimise the combined operation of the three lake system as it would result in huge memory and computing requirements. DDDP can, however, be used to optimise the operation of the system. The application of DDDP requires discretization of storage space for each of the lakes. The storage of Lake Victoria was discretized into 2000 levels. There are 80 and 91 data points for Lake Kyoga and Lake Albert respectively giving the levels in increments of 0.1 m. The corresponding values of the state variables are also available. The storage of Lake Kyoga and Albert were discretized in 80 and 91 levels respectively. To start the DDDP procedure, an initial feasible solution is required. The determination of such a solution is a cumbersome task because of the presence of negative NBSs in the historical record. For cases where the NBS are non-negative values, releases can be set equal to the NBSs to obtain an initial trial policy. However, for the Equatorial Lakes system the initial release policy was obtained by setting the releases to NBS if the NBS is positive. The releases are set equal to their lower bounds if the NBS is negative.

Starting with an initial solution, the DDDP procedure determines an improved trajectory within the specified corridor. One state below and one state above the trial state trajectory is used to form a corridor. The improved trajectory obtained in the previous iteration is adopted as the new trajectory to form a new corridor. The procedure is repeated for a number of iterations until there is no further improvement in the value of the objective function. At each stage of control horizon, the DDDP procedure gives the optimal lake levels, releases and power production at Lake Victoria. These sequences are then analysed to develop statistics of system performance. The real time operation procedure is similar to that followed for the single lake system. The forecasts of NBS to Lake Victoria for the next 12 time steps were made using the Thomas-Fiering model. For Lake Albert and Lake Kyoga, perfect forecasts have been assumed. The execution time for the DDDP model was around 400 minutes.

7.7.3.2 The GA approach

The GA was set up with tournament selection, elitism, uniform crossover and modified uniform mutation. For the Equatorial Lakes system, the chromosome representing the solution consists of 36 genes since there are 3 lakes in the system and the optimisation is carried out for 12 time steps. Since releases are the decision variables of the problem, each gene represents a release made from the lakes. For each time step of the historical record, the GA was run for 500 generations using a crossover probability of 0.7 and a mutation probability of 0.028 (1/36). A population size of 100 was used. After each run of the GA, the optimal levels in each lake, releases made from each lake, and power produced at different hydropower stations are recorded. The recorded data is then analysed to determine values of different parameters at various exceedence probabilities. The execution time for the GA model was 310 minutes which is less than for the DP model which required 400 minutes to execute.

7.7.3.3 Evaluation of model runs

The impact of real time operation of the three lake system on the levels in Lake Victoria is presented in Figure 7.30. The level sequences for Kyoga and Albert are shown in Figure 7.31, and Figure 7.32. It can be seen from Figure 7.30 that there is a good agreement between the levels produced at Lake Victoria by both the models. The sequences for Lake Kyoga and Albert clearly indicate that both the GA and the DDDP models achieve their aim of maintaining levels in the vicinity of mean historical levels. It can be observed from these sequences that there are periods in the control horizon when it is not possible to achieve levels close to the mean levels. This is largely due to the occurrence of very low or very high NBS in these periods. There is, however, a good correspondence between the GA and the DDDP results.

Figure 7.33 shows the power production at Owen Falls dam. Again, there is large fluctuation in power production similar to the one experienced for the single lake system. The fluctuating power production is not desirable from the operator's point of view as the contractual obligations generally require some minimum amount of power to be supplied each month. The situation where large amounts are generated during one month and less than the contracted amount in the next month needs to be avoided. With the DDDP model, the average power production is 146.9 MW and at 95% exceedence probability it is 64.5 MW. The corresponding values with the GA model are 146.5 MW and 63.8 MW. There is an improvement in the average power production with real time operation when compared to single zone regulation and the Agreed Curve. However, for the reasons mentioned in the previous section, the power production at higher reliability levels is low.

The case studies discussed in this section and the previous section indicate that with real-time operation it is possible to increase the overall energy generation from Lake Victoria but not the reliability of power production which is highly important for Lake Victoria. Real time operation should allow planning and forecasts of power production over a required duration of time. This is particularly important during periods of high lake levels.

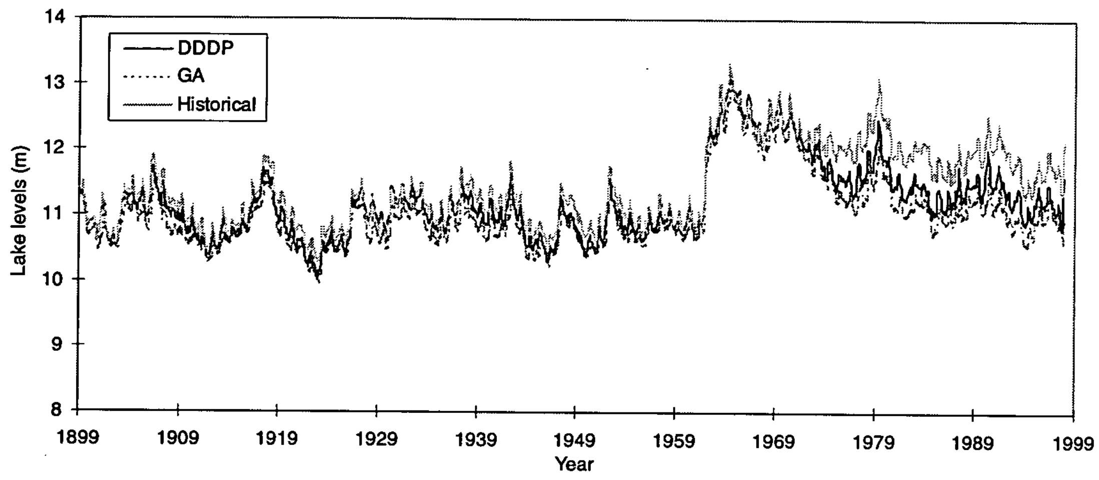


Figure 7.30 Lake Victoria levels, three lake system

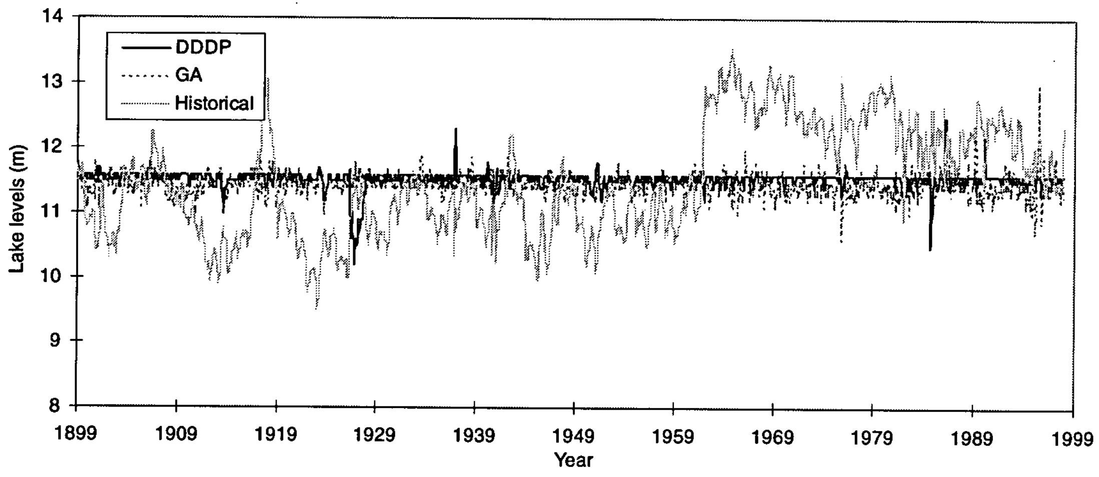


Figure 7.31 Lake Kyoga levels, three lake system

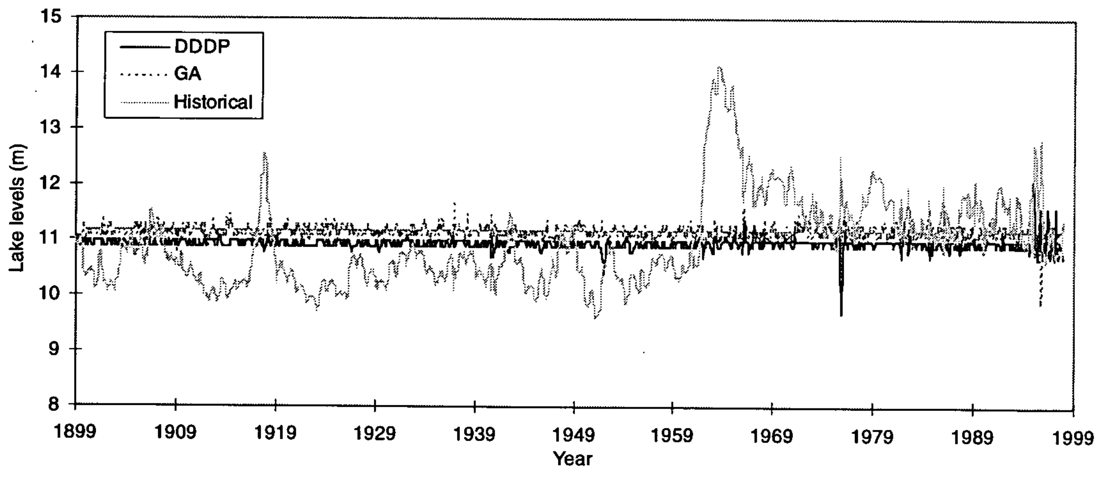


Figure 7.32 Lake Albert levels, three lake system

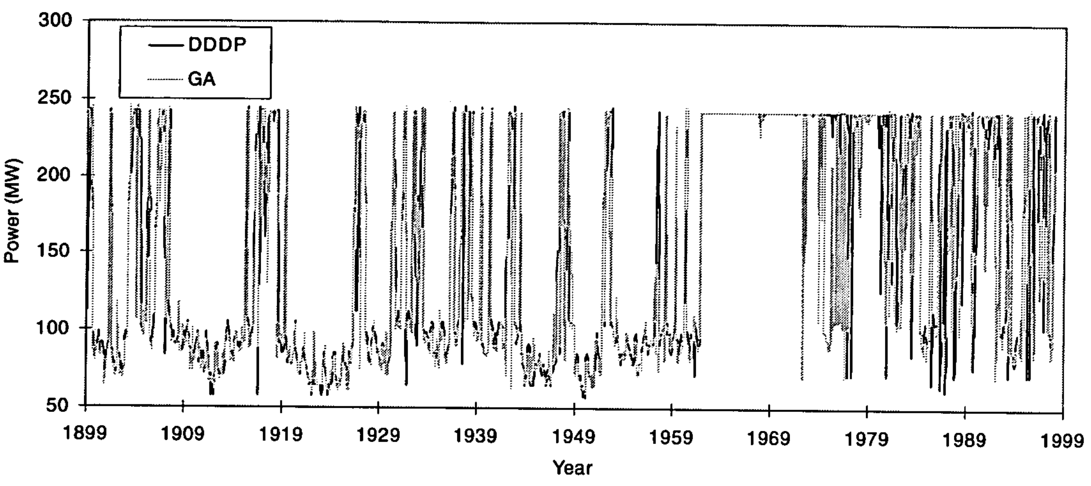


Figure 7.33 Power production at Owen Falls, three lake system

7.8 Power Forecasting Approach

This section presents a power forecasting approach (PFA) based on the GA for Lake Victoria. Most of the power systems are interconnected by hydro-thermal power systems. Although thermal plants are costly to operate, they are often installed in a relatively short time to fulfill a rapidly increasing demand. A disadvantage with thermal plants is that they cannot vary their generation from hour to hour with the flexibility of hydropower plants. Hydropower plants are much more responsive, and can be effectively used in supplying peak power as the generation rates can be varied rapidly. The choice of hydropower to generate peak power carries a higher economic value of water used. This makes optimal operation of any reservoir or lake in a hydro-thermal system a complex process because an operating decision in a given stage has a considerable effect on the consequences in the future. If higher amounts of hydropower are generated during the current stage, and low NBS occurs in the future, it may be necessary to use expensive thermal power in the future. On the other hand, if the lake levels are kept high through a more intensive use of thermal generation and high NBS occurs, there may be spillage in the system and an unnecessary waste of potential hydropower.

Studies of hydropower system operation typically involve determining policies to maximise total energy production. However, hydropower benefits accrue from capacity as well as energy benefits. Capacity benefits are based on the cost to construct thermal plants to replace reliable hydropower. The value of energy benefits is the fuel cost of replacing hydropower with thermal power. The objective of maximising the total energy production is a valid criteria in hydro-thermal systems with high installed thermal capacity. In systems where hydropower is a crucial part of the power infrastructure accounting for a high proportion of total installed capacity, concepts such as capacity benefits need to be taken into account in formulating management policies. In such systems, the reliability with which hydropower is produced is of utmost importance. At present, over 90% of the installed capacity in Uganda is from hydropower production at Owen Falls. Major future investments are also in hydropower installations on River Nile. With these new power projects, the potential to sell power internationally will be increased. Internationally sold power may be used as a substitute for existing thermal stations but planning of thermal power generation from these existing plants would need a high degree of reliability in hydropower production forecasts. It is therefore not sufficient just to maximise the hydropower production which may typically be the case in most hydrosystems.

The objective of regulating Lake Victoria must be to increase the reliability of power production. Any optimization procedure with the objective of maximising the total power production over a planning horizon would normally result in fluctuating power. Due to the manner in which the reliability of power production is calculated statistically, fluctating power production is bound to lead to low reliability similar to or even worse than that produced under Agreed Curve operation. Higher reliability in power production can be obtained if constant power is generated at every time step of the optimization horizon. The objective of generating constant power at every time step of the optimization horizon can only be achieved if the target power production is specified. However, determination of such a target is a highly complicated task. Moreover, it is a computationally complex task to incorporate capacity benefits in a DP formulation. To maximise reliable power output using a DP model, a series of optimisation runs with varying power targets would be required at every stage of the optimisation horizon. This would result in huge computational complexity and the benefits of optimisation might be offset by the computational costs. The traditional approach of determining reliable capacity of a hydropower system as described by Labadie et al. (1987) has been to use simulation models to determine the reliable power capacity of hydrosystems. The same approach has been outlined in the preceding sections of this chapter. The simulated output for a given operating policy is analysed to determine the reliable power at different risk levels. The uncertainty measure of the reliable power capacity can be obtained by repeating the procedure for different synthetic sequences.

It is important to be able to forecast the availability of minimum power that can be continuously generated from Lake Victoria for different durations of time particularly due to low installed thermal capacity in Uganda. In this research, a new alternative methodology for forecasting constant reliable power that can be produced over different durations of time such as 3 months, 6 months, 12 months and 24 months for a given forecast of NBS sequences, is presented. With these forecasts, it would be possible to satisfy more often the contractual obligations relating to supply of minimum reliable power. Also, it would be possible to plan the sale of any power produced in excess of reliable contracted power. Moreover, if the forecasts of power generation are such that reliable power is likely to be available for a particular duration, then shutting down the thermal power plant could be considered for that duration. This mode of operation would therefore allow savings in financial terms as well as provide an opportunity for maintenance and repair of thermal power plants.

The GA approach that would allow forecasts of reliable power over different duration of time is remarkably simple. A chromosome consisting of four genes is constructed with each gene representing a constant release that can be guaranteed for specified durations of time. A typical chromosome is shown in Figure 7.34, where G1, G2, G3, G4 represent the releases and G1 > G2 > G3 > G4. The first gene represents the constant release that can be guaranteed for the next 3 months while the second, third and fourth genes represent the constant releases that can be guaranteed for 6, 12, and 24 months respectively. A sketch showing the formulation is presented in Figure 7.35.

G1	G2	G3	G4

Figure 7.34 A typical chromosome used in PFA

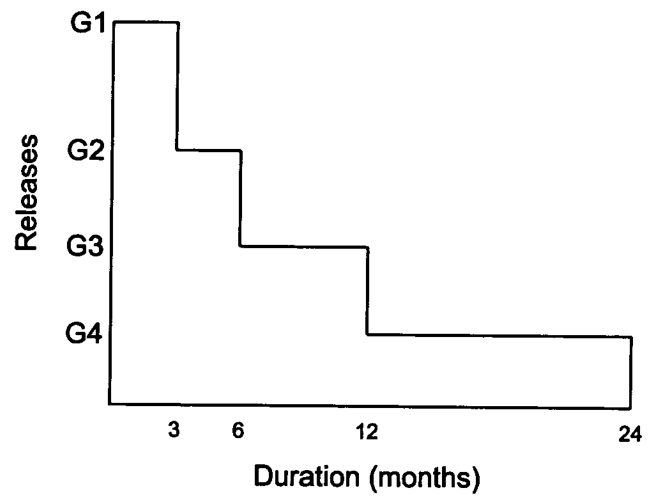


Figure 7.35 Formulation of the problem, PFA

The GA procedure begins with a random initialisation of a population of 100 chromosomes comprising of four genes each. The genes are assigned values between the lower and the upper bounds of releases. In the initialisation process, a random number generator that returned releases in fixed increments rather than continuous values was used. This speeds up the GA without incurring any loss in the accuracy of the solution obtained. A crossover probability of 0.7 and a mutation probability of 0.25 was used. The performance of the GA formulation described earlier was evaluated on the basis of a design NBS sequence with a 90% probability of exceedence. This sequence was derived by carrying out critical period analysis on a synthetic sequence generated by the Thomas-Fiering model, and is shown in Figure 7.36. For each month in the simulation horizon, a forecast based on design NBS sequence is made for the next 24 months. The GA process was run for 100 generations, and at the end of each run an optimal or a near optimal sequence of releases that can be guaranteed for 3, 6, 12, and 24 months is obtained. From these sequences, only the first three month's releases are actually implemented. The state of the lake is updated with actual NBSs because the forecast NBS usually deviate from the actual NBS. The starting level for the next control-simulation time step is determined from the results of the previous time step. After the first 3 months releases have been implemented, the model is run again to derive a new release policy for the next 24 months. This new release policy is determined taking into account the releases that have been guaranteed for different durations during the previous time step of the model run. This control-simulation process is repeated for the entire historical record in steps of 3 months. The total execution time for the GA model was 40 minutes.

The reliability with which the power forecasts are met is also important. As the model is run in steps of three months, the releases guaranteed for the next three months during the current stage are implemented in the current stage as well as in the next two stages. This implies that the 3 month's forecasts of power production are met with 100% reliability. The model has an imbedded constraint which require that the forecasts of 3 month's releases made in the next run of the model should not be less than the release that have already been guaranteed for the next 6 months during the current run of the model. In this way, the model has a constraint such that the power forecasts for upto next 12 months should not be less than that already guaranteed during the earlier model runs. However, no such constraints could be set up on 24 month's forecasts. Setting up such a constraint on 24 month's forecast would imply that these forecasts could not be decreased in subsequent model runs; they could only be increased which is practically not possible.

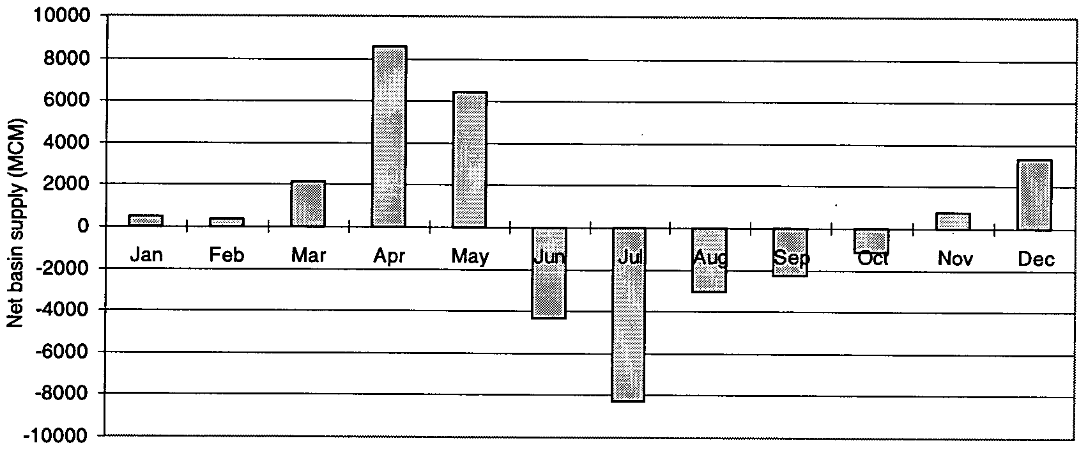


Figure 7.36 Design NBS sequence for Lake Victoria

The PFA described in the preceding sections was also evaluated using the perfect forecasts which were based on the historical NBS. These two control-simulation experiments resulted in sequences of lake levels, releases, power production in each time step of the historical record. A statistical analysis was then carried out on these sequences to evaluate the performance of the system. Figure 7.37 presents the comparison between the historical levels and the levels produced by the GA model using the critical NBS sequence at 90% probability of exceedence. The levels follow a pattern similar to one observed under the Agreed Curve operation particularly prior to the periods of high lake levels. During periods of high lake levels in the historic record, the model produces high amounts of power and consequently the lake levels go down as seen in Figure 7.37. The minimum level was 10.0 m which is 0.22m lower than the historically observed minimum level. The water levels for Lake Kyoga and Lake Albert are shown in Figure 7.38 and Figure 7.39 respectively. There is no significant impact on the levels in these lakes.

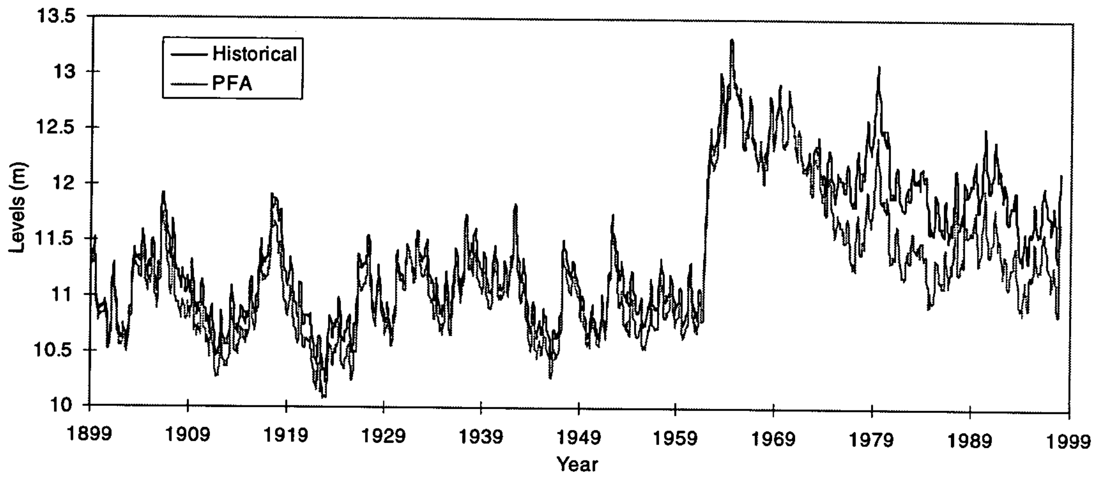


Figure 7.37 Lake Victoria levels, power forecasting approach

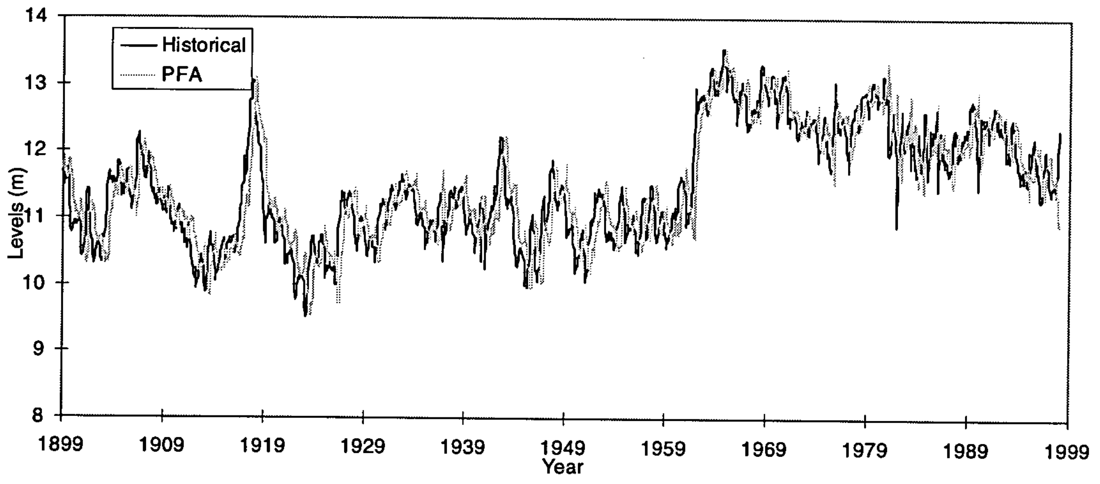


Figure 7.38 Lake Kyoga levels, power forecasting approach

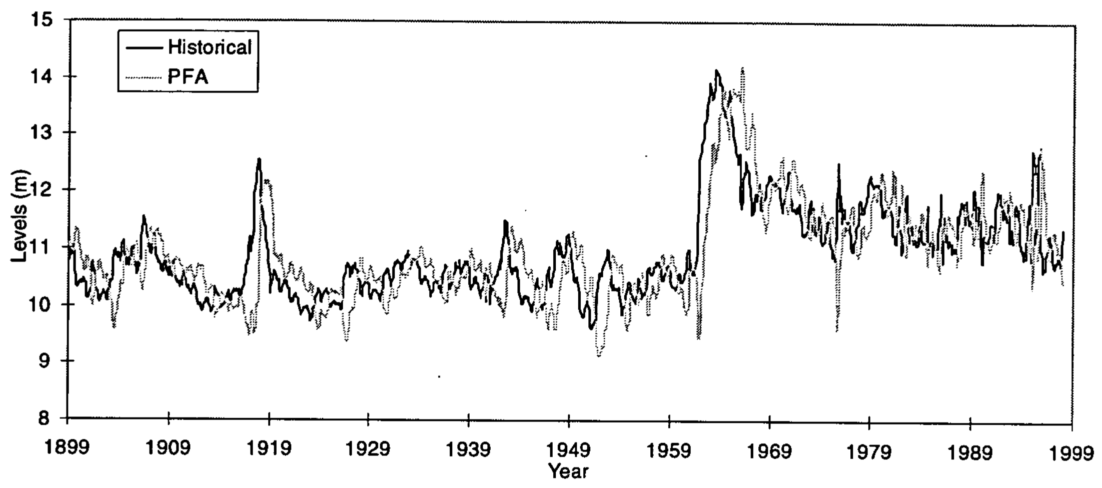


Figure 7.39 Lake Albert levels, power forecasting approach

The impact of regulation considered in this section on Lake Victoria discharges is shown in Figure 7.40. There is a distinct impact on the discharges. During periods of high lake levels, higher discharges are maintained thus giving stable power production for longer periods. However, there is large variability in the discharges when compared to the single zone regulation except during periods of high levels. The flow duration characteristics from the Owen Falls for the 3 months forecasts are presented in Figure 7.41.

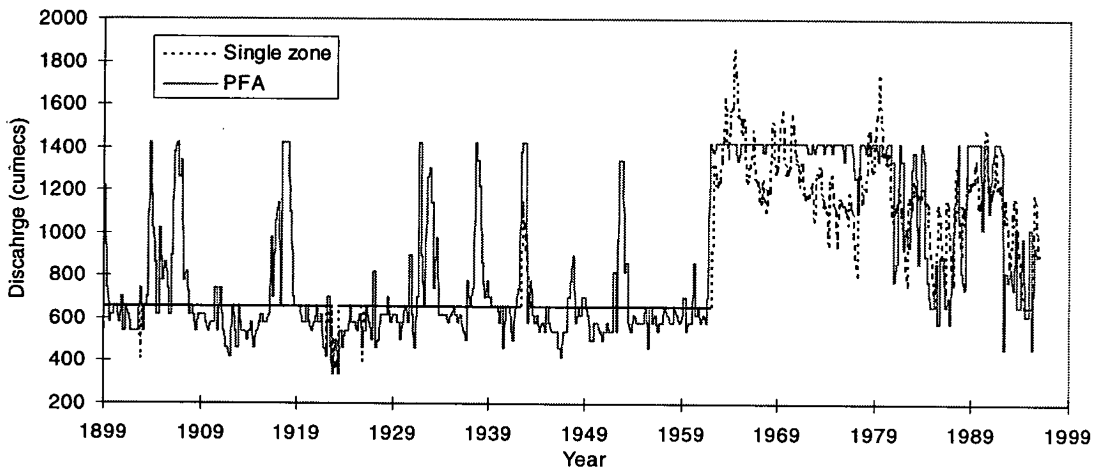


Figure 7.40 Lake Victoria discharges, power forecasting approach

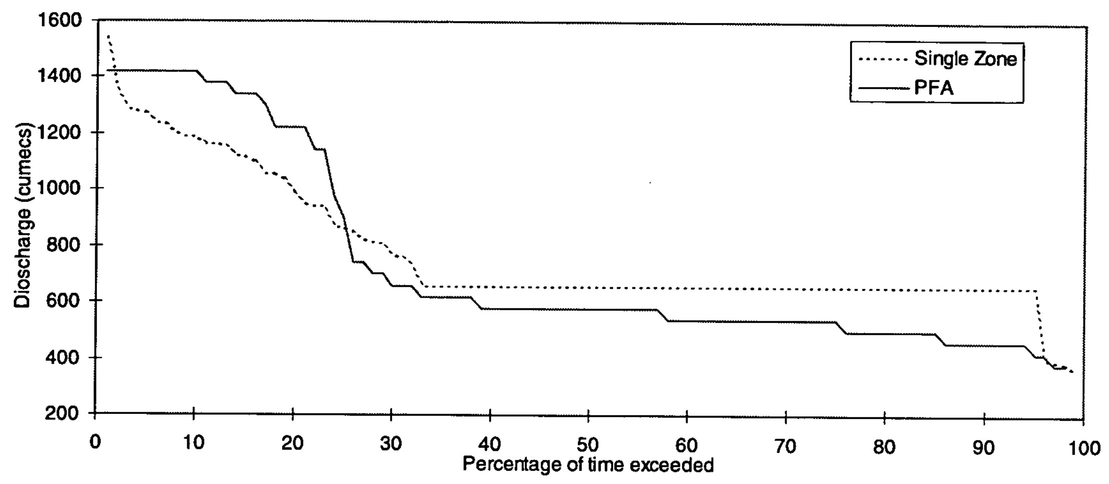


Figure 7.41 Owen Falls flow duration curve, 3 months forecasts

The power production from the Owen Falls dam is shown in Figure 7.42. There is large variability in power production with the pattern being very similar to the one observed for the discharges. It can be seen from Figure 7.42 that the model makes use of the high lake levels by maintaining higher constant power for longer period.

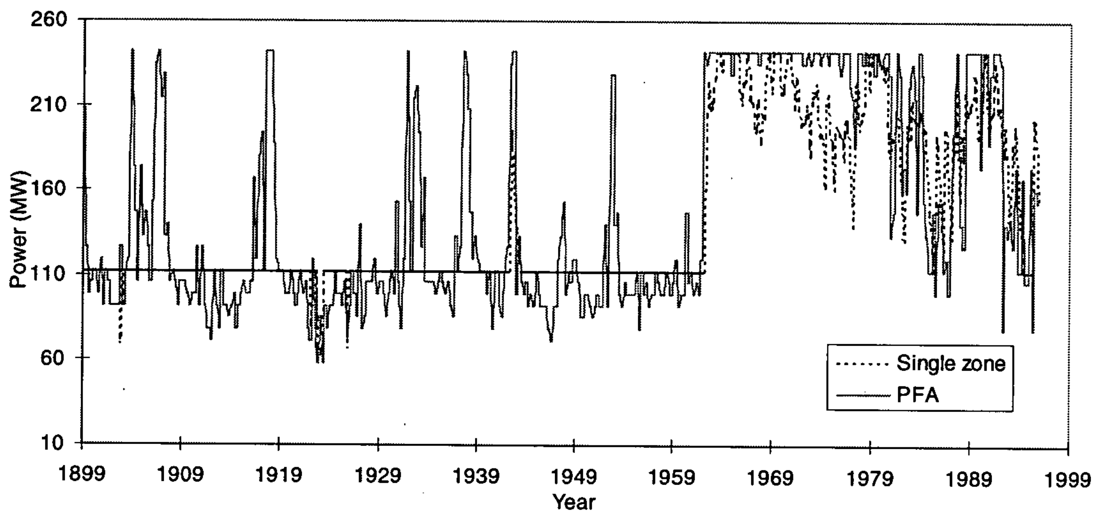


Figure 7.42 Owen Falls power production, 3 months forecasts

Table 7.5 gives the summary of power production at Owen Falls and other installations downstream of Owen Falls. The average power production increased from 143.9 MW under single zone regulation to 148.2 MW under PFA. The discharge that can be met with 95% annual reliability is 460 m^3/s and the corresponding reliable power output is 78.4 MW which is significantly higher than is possible under the Agreed Curve operation where the reliable power production was 72.5 MW. The increase in reliable power production is not to the same extent as that obtained under single zone or the dual zone regulation but the main advantage of the type of regulation considered here is that allows forecasting of power production at different planning horizons. The power duration curves for 3 months, 6 months, and 12 months forecasts are shown in Figure 7.43, Figure 7.44, and Figure 7.45 respectively.

Table 7.5 Summary of power production in MW, PFA

Installation	**Power Forecasting Approach**				**Single Zone**	
	Critical NBS Series		**Perfect Forecasts**		**Historical NBS**	
	Average	**95%**	**Average**	**95%**	**Average**	**95%**
Owen Falls	148.2	78.4	148.7	78.4	143.9	112.5
Bujagala	122.43	69.4	125.5	69.4	121.2	98.0
Kalagala	208.11	114.9	209.9	114.8	204.3	163.1
Kamdini	126.1	81.2	126.6	81.2	129.3	90.0
Ayago	223.1	166.9	223.1	166.9	225.9	184.8
Murchison	313.3	279.1	313.3	279.1	314.5	308.4

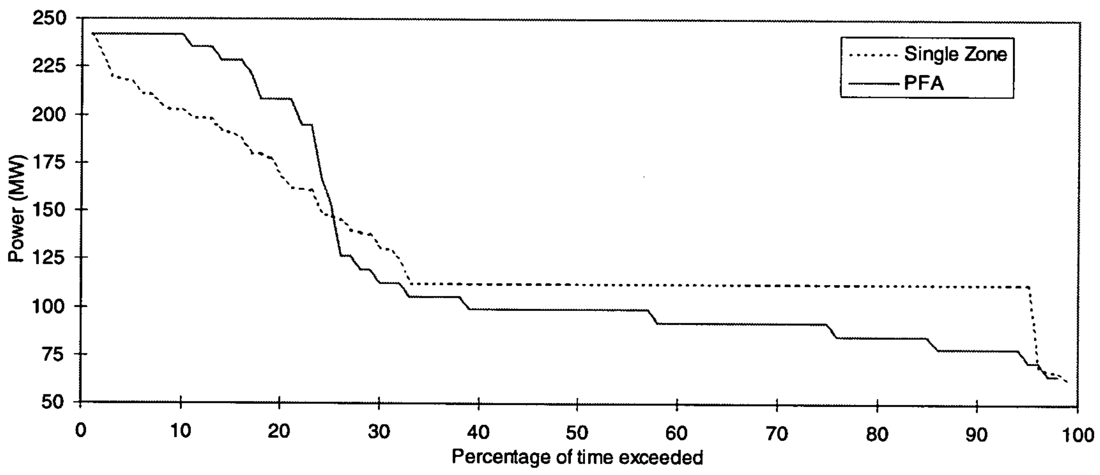


Figure 7.43 Owen Falls power duration curve, 3 month forecasts

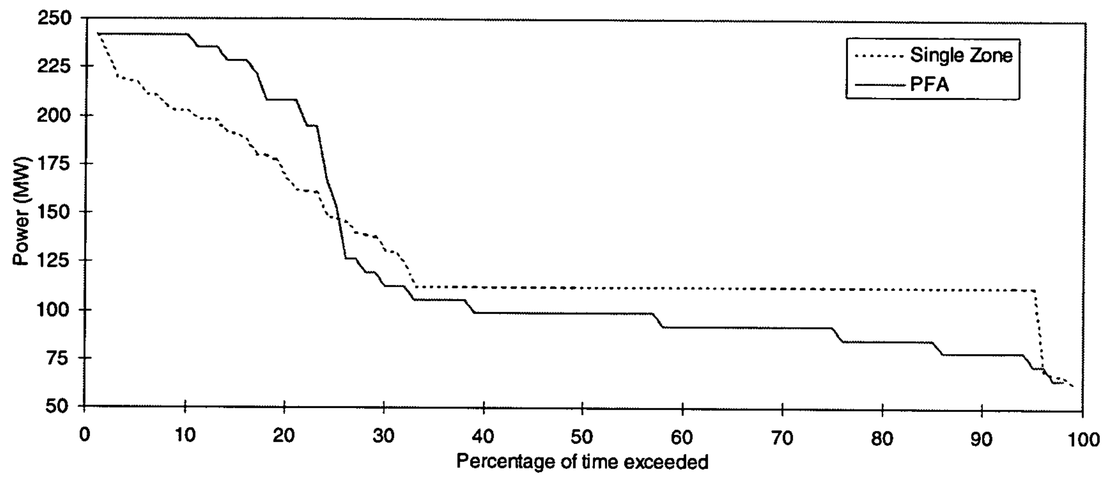


Figure 7.44 Owen Falls power duration curve, 6 month forecasts

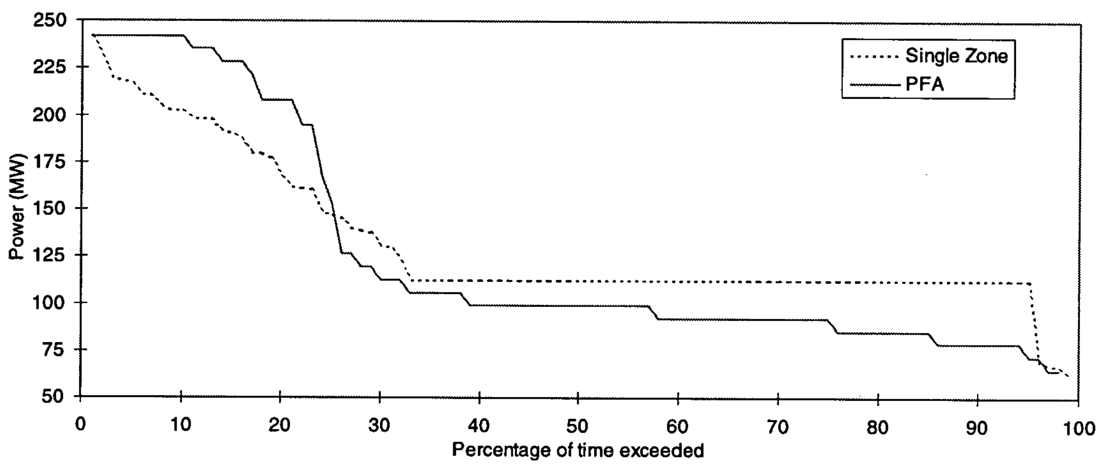


Figure 7.45 Owen Falls power duration curve, 12 month forecasts

While not providing capacity benefits, the mode of operation described in this section can offer higher overall returns. With the forecasts of potential power production available, it would be possible to plan the sale of surplus hydropower as a substitute of thermal power. To investigate this, a selling price of 0.05 US $/KW-h for the contracted power was assumed. The energy benefits from selling the surplus hydropower cannot be treated in the same way as the capacity benefits that derive from reliable power production. Therefore, a power tariff structure was also assumed. The return from the power that can be guaranteed for the next 3 months was assumed to be only 60% of that obtained from selling the reliable power. Similarly, the returns for the power that can be guaranteed for the next 6 months and next 12 months were assumed to be 75% and 90% of that obtained from the reliable power respectively. An economic analysis was then carried out to determine the total returns that can be obtained from the system under different operating procedures.

Based upon the power tariff structure described above, the total economic return from the power production during 99 years of operation under single zone regulation was 4791.083 million US $. This value of economic return was obtained considering that with single zone regulation there are no extra benefits available for producing power in excess of the contracted power. However, with the potential power forecasting approach, it would be possible to sell the surplus power. The total economic return using 3 months power forecasts was 5196.059 million US $, an increase of around 404.976 million US $ over the single zone operation. The sensitivity of this increase in benefits to the assumed power tariff structure was also investigated. It was found that a long as the power guaranteed for the next months could be sold at 47% or more of the price of the reliable power, there will be an increase in the benefits obtained with the power forecasting approach. The summary of economic analysis is presented in Table 7.6.

Table 7.6 Summary of economic benefits in million US $

Forecast Duration	Critical NBS Series	Perfect Forecasts
3 months	5196.059	5210.985
6 months	5812.744	5830.203
12 months	6121.050	6139.773

The benefits from operation of Lake Victoria could be further enhanced using a combination of PFA and single zone regulation. The Lake could be operated according to single zone when the levels are below a particular threshold and according to PFA if the levels are above the threshold. Determination of the threshold would require experimentation, and values between 11.0 and 12.0 m should be considered. With this type of operation, the reliability of power production may be improved while capitalising on higher lake levels whenever they do occur.

7.9 Conclusions

In this chapter, the impact of different operating procedures on the levels, discharges and power production from the Equatorial Lakes was considered. With improved regulation schemes like the single zone and the dual zone regulation, significant increase in the reliable power production from Owen Falls was obtained. All subsequent hydropower projects downstream of Lake Victoria would also benefit from regulation. Both the single zone regulation and dual zone regulation have a significant impact on discharges in the section of the River Nile between Lake Victoria and Lake Kyoga. The extremes of the Lake Victoria levels and discharges simulated with these regulation schemes are within the range of those experienced under Agreed Curve operation. The regulation has very little impact on the levels and discharges from Lake Kyoga and Lake Albert. The performance of single zone regulation with a 990 year long synthetically generated NBS sequences was also highly satisfactory.

The real time operation of Equatorial Lakes system was also considered. Models based on the DDDP and the GA approach were developed for the single lake system and the three lake system. The results obtained using the GA approach were similar to those obtained by the DDDP approach which demonstrates further the robustness of the GA approach for real time operation of multi reservoir systems. However, the reliability of power production is low with both the GA and the DDDP models. This is largely due to the extreme variability in power production experienced with real time models having the objective of maximising the overall power production. Such an operation would not be suitable for Lake Victoria where reliability of power production is very important due to low installed thermal capacity.

A GA model for forecasting reliable power that can be produced over different durations of time has also been developed. The 3 month's forecasts of power production are met with 100% reliability, and there are constraints in the model which require that the power produced in any month should not be less than that already forecast for that particular month. The PFA approach resulted in an increase in the reliable power over that possible with the historical operation but not to the same extent as with the single zone operation. The main advantage of this approach, however, is that the sale of surplus power is possible. The results of the economic analysis clearly indicated that if reliability is less of a consideration because of higher installed thermal capacity, the power forecasting approach will provide better returns than either the Agreed Curve or the single zone regulation. Depending upon the forecasts of potential power production, decisions can be made to cease the generation of expensive thermal power. This would also allow maintenance and repair of thermal power plants particularly during the period of high lake levels. It may be concluded that the power forecasting approach would also make use of the higher flows when they are available as they have been over the past 35 years. Also, it would be possible to plan the operation of thermal plants if the power forecasts from the hydropower plants are available. Furthermore, the GA formulation developed herein lends itself to operational applications without much modification.

8. CONCLUSION

8.1 Introduction

The research carried out in this thesis discusses the implementation of GAs for the optimisation of multi reservoir systems. The application of GAs to a number of problems has been presented and their effectiveness in identifying optimal or near optimal solutions has been analysed. Wherever possible, the GA performance has been evaluated against other techniques. This chapter summarises the research reported in this thesis, outlining the limitations of the research and providing recommendations for future research.

Following this introduction, section 8.2 presents the achievements of the research. Section 8.3 describes the limitations and section 8.4 presents a few pointers towards the future work.

8.2 Achievement of the Thesis

The research presented in this thesis is made up of three distinct parts. The first part of the thesis presents the development of a GA for water resource problems and investigates its performance on a number of problems with known solutions. These applications offered the opportunity to verify the results produced by the GA. Secondly, the approach was applied in the optimisation of control curves for a multi reservoir system in Indonesia. Finally, application of the GA technique in investigating different operating procedures for the Equatorial Lakes system in Africa is described. Real time operation of the system was also considered, and a method of reliable power forecasting developed with the GA. Major achievements of the thesis highlighting the contributions at each stage are presented below.

Chapter 2 of the thesis, presents a review of optimisation techniques for reservoir systems operations, with particular emphasis on techniques that aimed at alleviating the dimensionality problem associated with the DP approach. Chapter 3 of the thesis presents an introduction to GA technique, and a brief review of GA applications to problems from different disciplines of civil engineering. The GA jargon is discussed in detail and the procedure for formulating a problem for solution by GA is outlined in its general form. This could serve as a useful guideline in the implementation of GAs to optimisation problems from diverse fields.

In Chapter 4 of the thesis, an evaluation of several alternative formulations of the GA for the well known four reservoir problem is presented. A sensitivity analysis of GA performance to parameters such as mutation probability, crossover probability, and population size is also carried out. Different representation schemes and reproduction operators are also investigated. The application to a non-linear four reservoir problem, and to a problem with extended time horizons has further demonstrated the robustness of the approach. In all cases, the GA was able to identify near optimal solutions. Some major achievements from this chapter are summarised here.

- A procedure for formulation and implementation of GAs to multi reservoir optimisation problems has been developed.
- A generic GA model for the optimisation of reservoir systems has been developed which is transportable with minimal changes to any reservoir system.
- Guidelines have been laid for determination of GA parameters that provide the best performance.
- A best set of reproduction operators for a GA has been identified.
- The effectiveness of real-value representation over binary and Gray representation has been demonstrated.
- A complex ten reservoir problem has been successfully solved.
- Application to a four reservoir problem with extended time horizons has also been demonstrated.

In Chapter 5 of the thesis, the characteristics of the Brantas Basin and the essential features of the simulation model of the Basin are described. Chapter 6 presents the application of the GA approach in the optimisation of control curves for the existing development situation in the basin, and also for two future water resource development scenarios. A comparison of the GA results with those produced by DDDP is also presented. For each case considered in this chapter, the GA results are very close to the optimum and the technique appears to be robust. Contrary to methods based on DP, discretization of state variables is not required. Further, there is no requirement for trial state trajectories to initiate the search using a GA.

The computational complexity for a GA increases at a slower rate than that for the DP based methods. Some major contributions from this chapter are listed below.

- Control curves which can be used in the actual operation of the reservoirs have been developed for the existing development situation in the basin, and also for two future water resource development scenarios in the basin.

- Rule curves with improved benefits in terms of irrigation production from the Brantas delta have been developed.

- It has been demonstrated that GAs have considerable flexibility in application to problems with complex objective functions.

- GAs generate a number of alternative solutions close to the optimum solution providing more choice to the operator of the reservoir system.

In chapter 7 of this thesis, a number of regulation plans for Lake Victoria are considered with the objective of improving the reliability of power production. The application of the GA approach in the real time operation of the Equatorial Lakes system is also investigated. A power production forecasting approach using the GA has also been developed that allows power production to be forecast over different durations of time. Such an operation could not have been achieved using DP. The approach was found to be economically superior to the single zone regulation under the assumption that reliability is less of a consideration because of high installed thermal capacity. The major conclusions from this chapter are summarised here.

- The single zone and the dual zone regulation provide around 50% increase in the reliability of power production from Lake Victoria over the current operation.

- Construction of thermal power plants can be avoided in the short to medium term if Lake Victoria is regulated according to single zone or dual zone regulation.

- With single zone regulation, an increase of 30% in the reliable power, which amounts to 194 MW, can be obtained if all the subsequent hydropower projects downstream of Owen Falls are implemented.

- Neither the single zone nor the power forecasting approach require sophisticated operational controls and could be easily implemented.

- A combination of single zone and power forecasting approach could lead to enhanced benefits.

In the context of reservoir optimisation, GAs have some distinct practical advantages over the traditional algorithms. Complex and non-differentiable objective functions that frequently occur in reservoir operation problems introduce no difficulties as a GA requires objective function or fitness information only. Discretization is a major cause of dimensionality problem associated with DP based methods but in a GA, state and decision space need not be discretized. Finally, implementation of GA is straightforward due to its simplicity and ease of interfacing.

8.3 Limitations of the Research

This section presents the limitations of the research presented in this thesis.

GAs are a heuristic search technique and are not theoretically guaranteed to find the optimal solution to a given problem. This is not particularly important for complex real world reservoir systems as these are already being managed with operating policies that are based upon experience or are derived by some approximate optimisation technique. Although GAs are not guaranteed to identify the exact optimum, they maintain a multipoint perspective on many regions of the solution space simultaneously and thus have a high probability of locating the true optimum.

The GA approach requires the use of penalty functions to satisfy the constraints of the problem. Proper settings of penalty parameters is vital to achieve an efficient performance from a GA. Too high penalty parameters are likely to eliminate potentially good solutions at an early stage of the run whereas if the penalty parameters are low, bad solutions may propagate in subsequent generations. The problem of constraint satisfaction can be severe if very long chromosomes are considered. The use of appropriate penalty functions can overcome the problem but some experimentation might be required. The problem of constraint handling is, however, not limited to the GA approach; other approaches such as optimal control theory also face the same problem. While DDDP can easily handle the constraints on state and decision variables the method is not efficient for large systems in terms of execution time and memory requirements. On the other hand optimal control theory approaches are highly efficient in terms of execution times but their application is limited to problems with continuous and differentiable objective functions.

The other limitation of the GA approach is that they are controlled by a number of parameters and their success largely depends upon the proper setting of these parameters. The problem is that no single set of parameter values will result in an effective and efficient search in all cases. For this reason, a sensitivity analysis to GA parameters has been carried out in this thesis which clearly indicates that the mutation probability affects the performance of a GA. The results of sensitivity analysis with respect to crossover probability are stable and a value of around 0.7 has been found to be most suitable. The population size also affects the GA performance but not to the same extent as the mutation probability. Hence, a correct choice of mutation probability is important, and some care has to be exercised in selecting the values of mutation probability. The guidelines laid down in this thesis suggests that a mutation probability corresponding to one mutation per chromosome on an average may be optimal. The initial tests of GA performance should be carried out with values of mutation probability corresponding to around one mutation per chromosome.

8.4 Recommendations for Further Research

To conclude the thesis, the following recommendations are made for further research which could lead to enhanced optimisation performance.

8.4.1 Hybrid GAs

The first suggested area in which the research can be undertaken follows from the limitation of the GA in locating optimal solutions. A hybrid search strategy that combines GA with a local search method could produce more efficient search algorithms. The basic GA is effective in identifying solutions close to the optimum because it performs a global search instead of a localised one. The solution achieved by GA could then be improved using a local search technique. For example, simulated annealing could be used to refine the solutions already found by a GA. Franchini (1996) used sequential quadratic programming as a local search method to improve the solutions produced by the GA. Applications of GA in combination with fuzzy logic have been described by Lee and Takagi (1993) and Soh and Yang (1996). Many applications of hybrid GAs have been reported by Davis (1991).

8.4.2 Improved GAs

The performance of classical GAs can be improved through the introduction of advanced operators. Numerous versions of selection schemes, crossover and mutation operators are being devised by the researchers in the pursuit of further improving the performance of GAs (Michalewicz 1992; Dasgupta and Michalewicz 1997). One of the possible directions for research is investigation of the effect of more sophisticated reproduction operators on the performance of GAs. In this thesis, GAs with constant population size are considered. It would be interesting to implement and analyse the performance of GAs in which the population size varies with the evolution process.

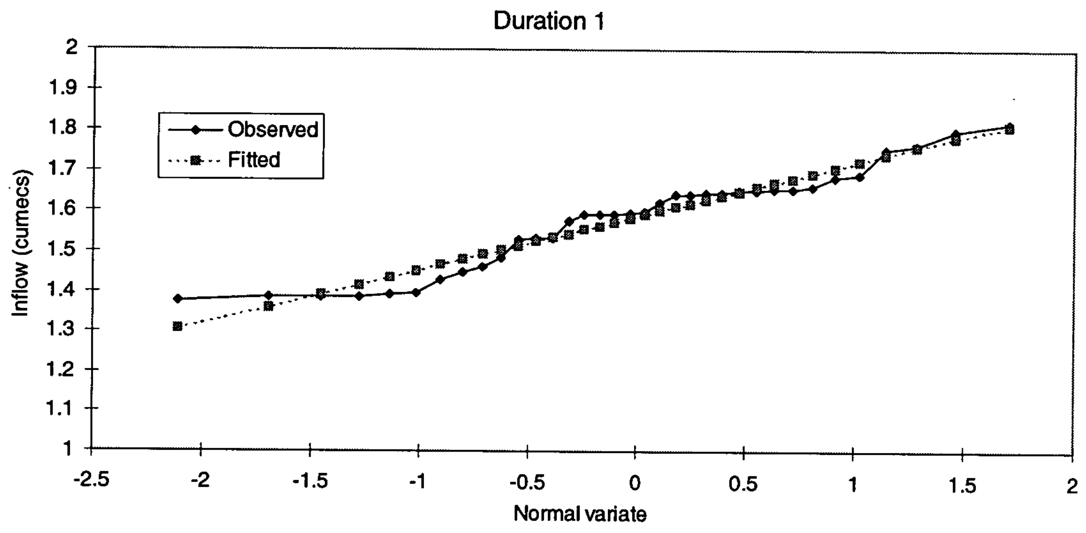
8.4.3 Parallelisation

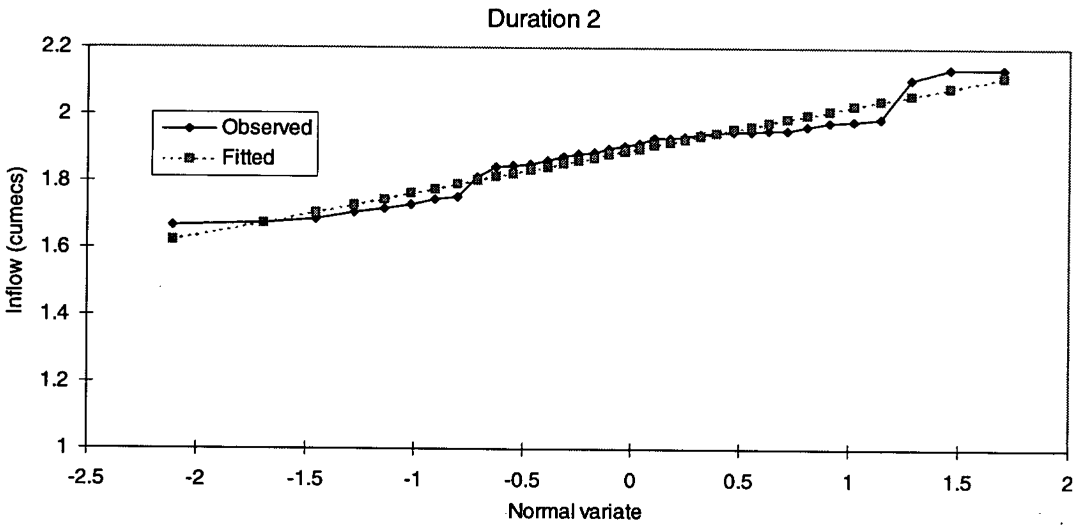
One of the major advantages of GAs is their ability to be parallelised. Natural evolution in itself is a highly parallel process as it deals with a population of individuals and not with a particular individual. Parallel GAs work by distributing the task of a basic GA on different processors. Since each chromosome can be evaluated independently of the other, significant savings in execution time could be achieved. With parallel GAs it is possible to use higher population sizes, reduce the computing time, and thus improve the overall performance. The parallelisation process cannot be used for techniques such as LP, DP or DDDP because of the manner in which these methods work. On the other hand, GAs are fairly straightforward to implement on parallel machines. The speedup attainable by solving GAs on parallel computers could be significant for problems where the time required to evaluate each solution is high.

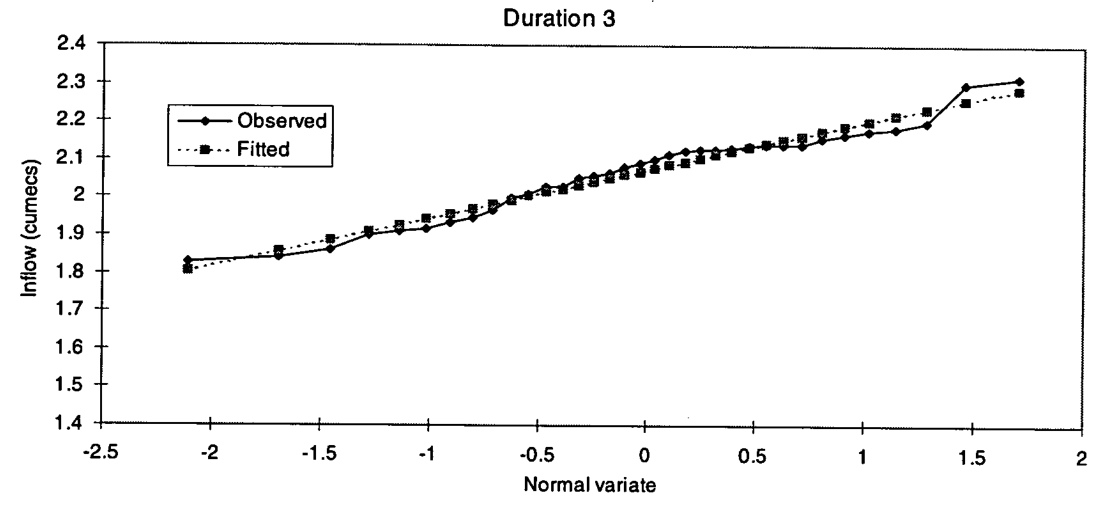
Appendices

A. Probability Distribution of Inflows to Karangkates

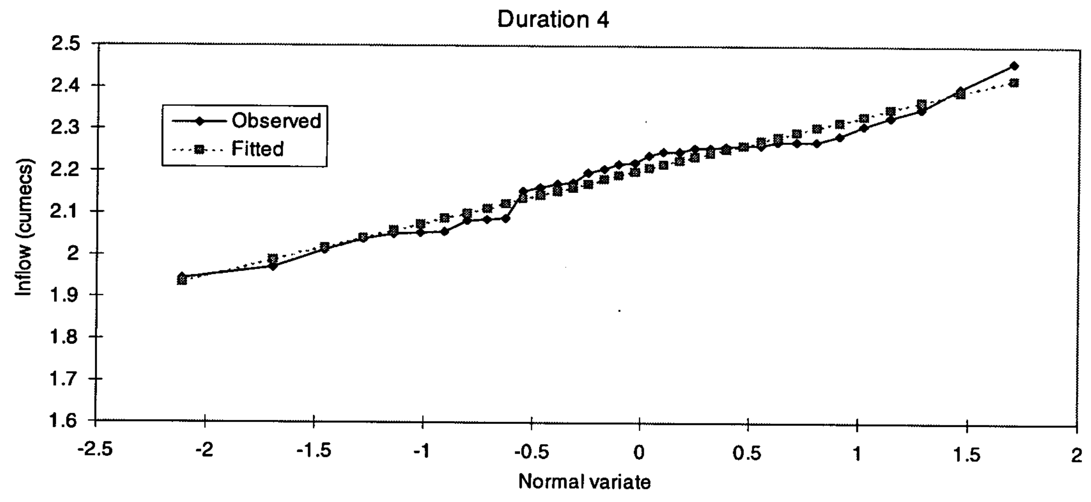
The following figures show the log-normal distribution fitted to the series of 10-day inflows to Karangkates reservoir. It can be observed that the log-normal distribution fits the data well for most of the durations. The values shown on the y axis are the logarithms of inflows in different durations.



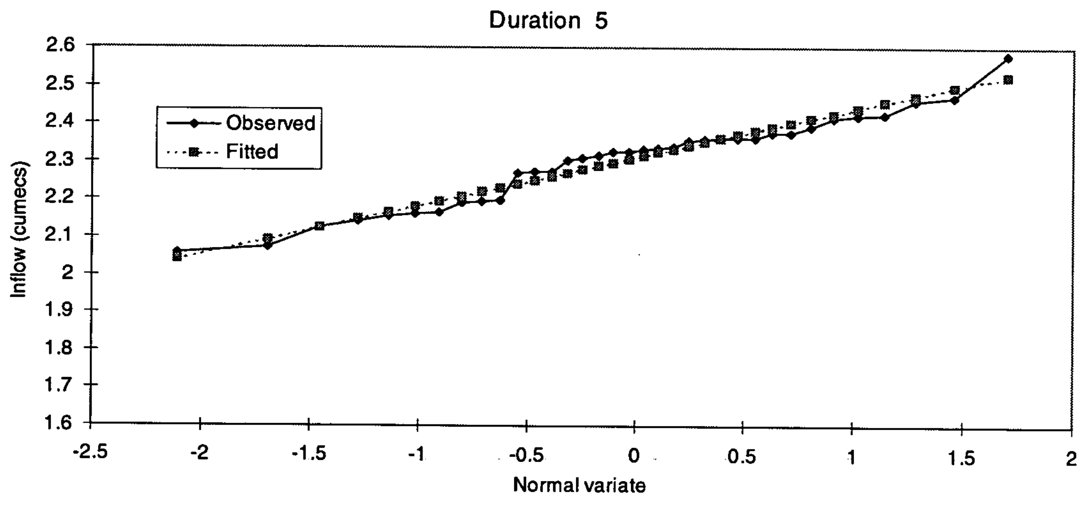




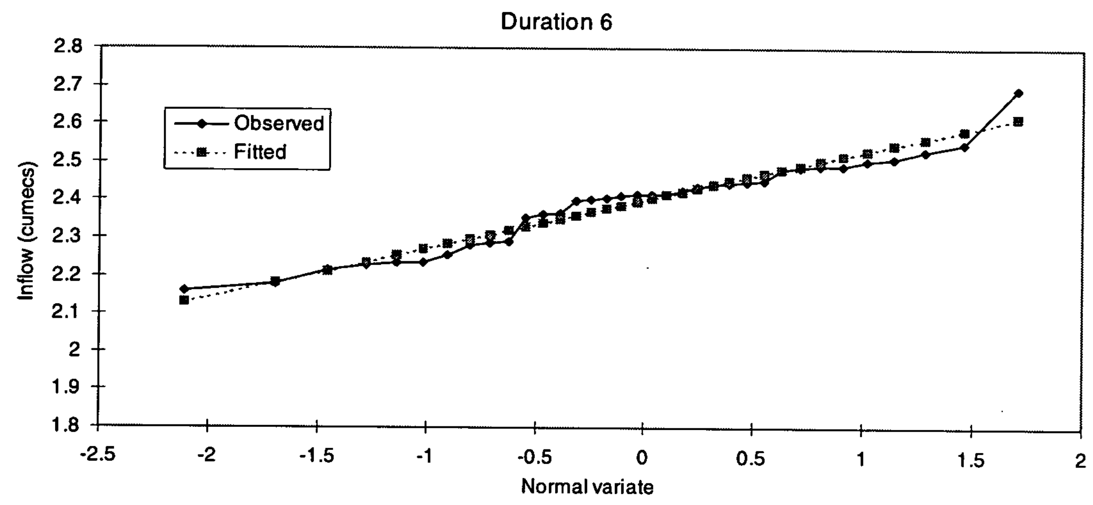
Duration 3
Inflow (cumecs)
Normal variate
Observed
Fitted
2.4
2.3
2.2
2.1
2
1.9
1.8
1.7
1.6
1.5
1.4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

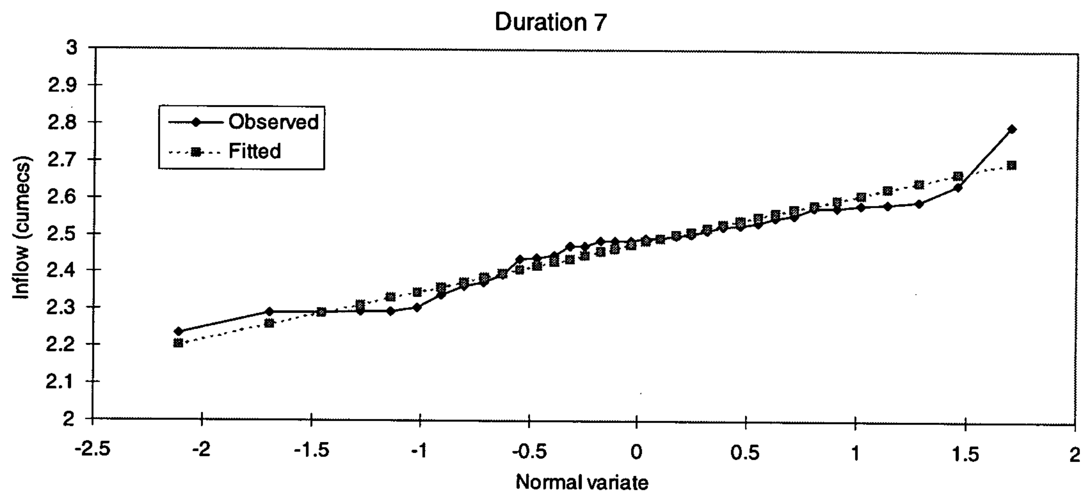


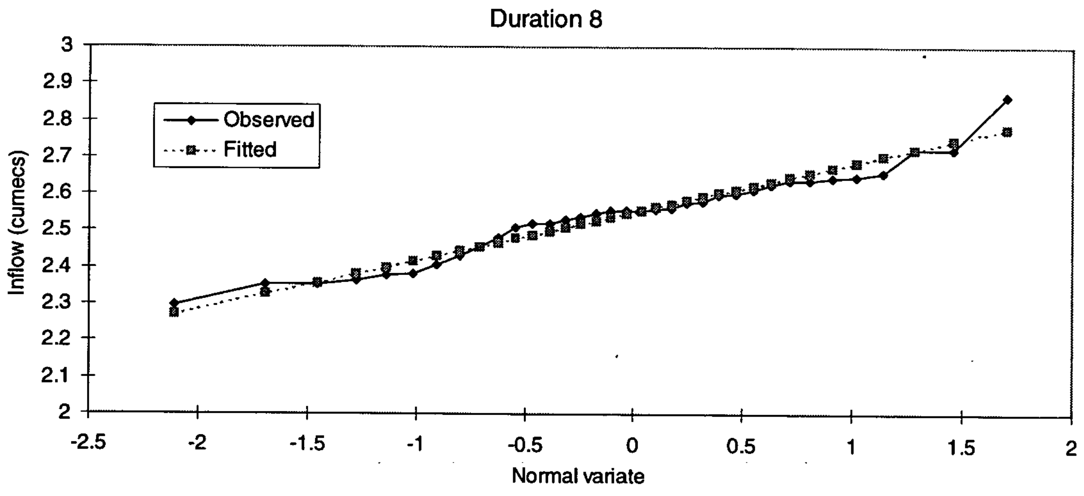
Duration 4
Inflow (cumecs)
Normal variate
Observed
Fitted
2.5
2.4
2.3
2.2
2.1
2
1.9
1.8
1.7
1.6
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

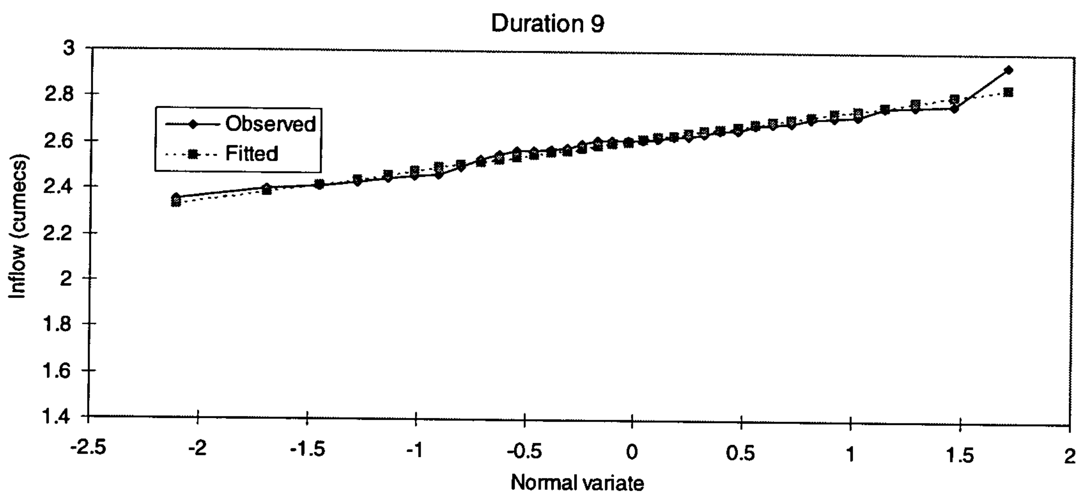


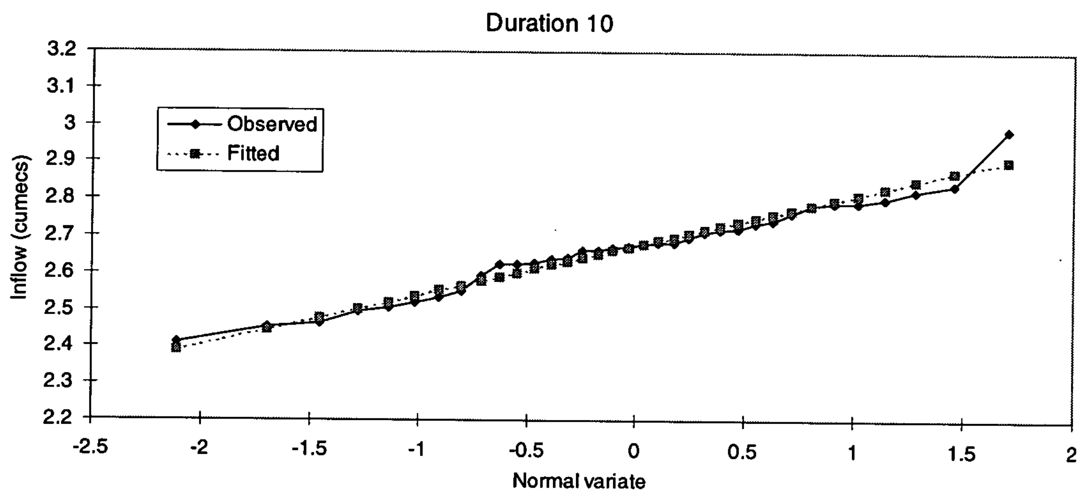
Duration 5
Inflow (cumecs)
Normal variate
Observed
Fitted
2.6
2.5
2.4
2.3
2.2
2.1
2
1.9
1.8
1.7
1.6
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

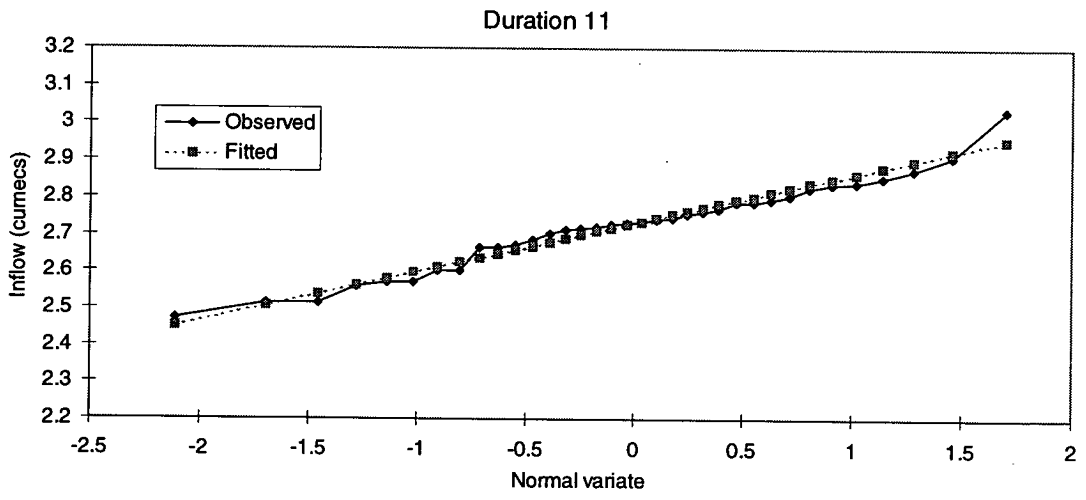

Duration 6
Observed
Fitted
Inflow (cumecs)
Normal variate
2.8
2.7
2.6
2.5
2.4
2.3
2.2
2.1
2
1.9
1.8
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

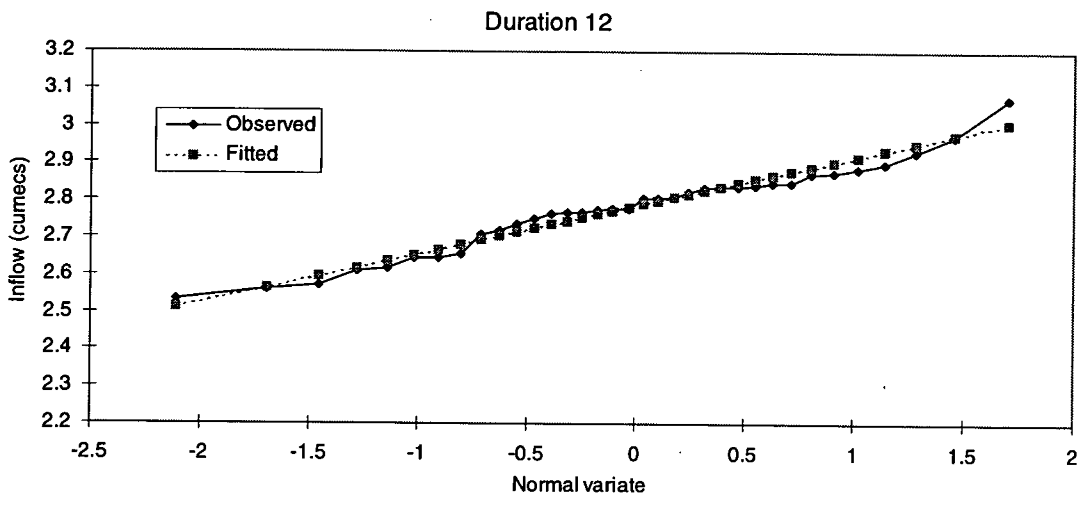

Duration 7
Observed
Fitted
Inflow (cumecs)
Normal variate
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
2.1
2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

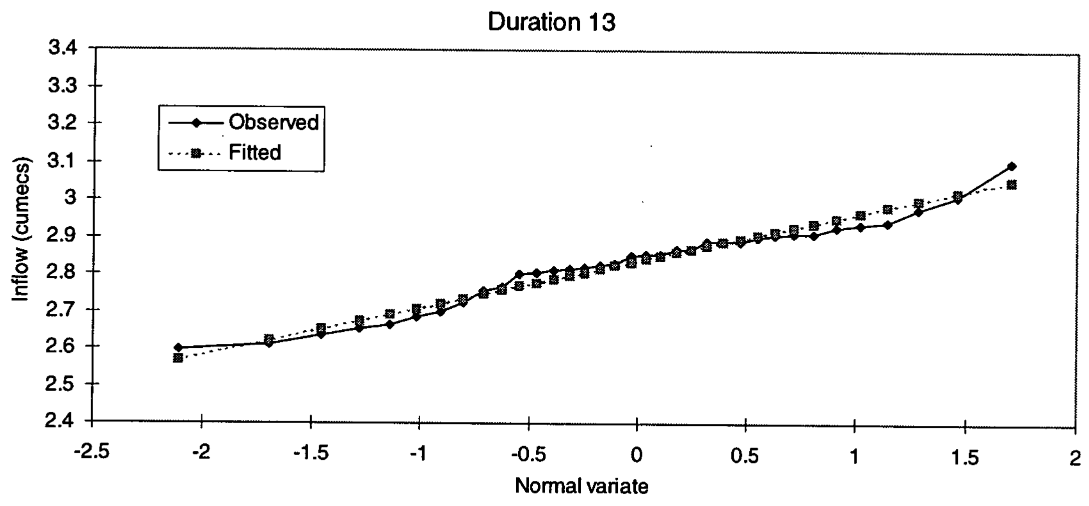

Duration 8
Observed
Fitted
Inflow (cumecs)
Normal variate
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
2.1
2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

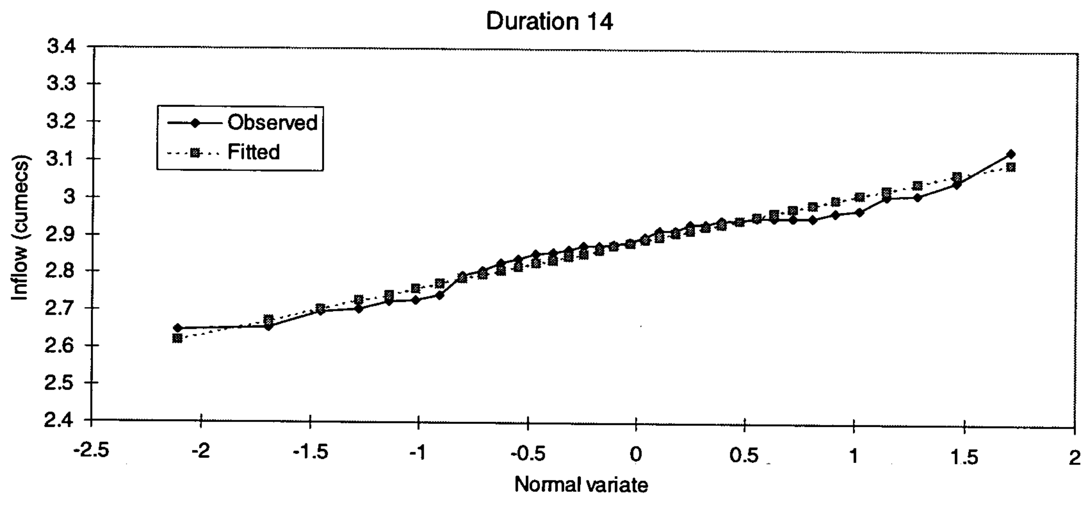

Duration 9
Inflow (cumecs)
Normal variate
Observed
Fitted
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

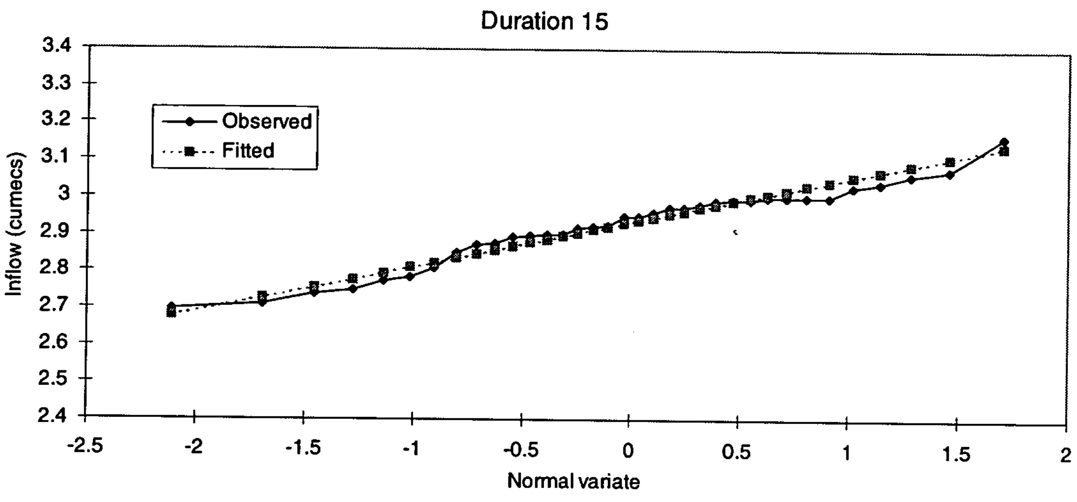

Duration 10
Inflow (cumecs)
Normal variate
Observed
Fitted
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

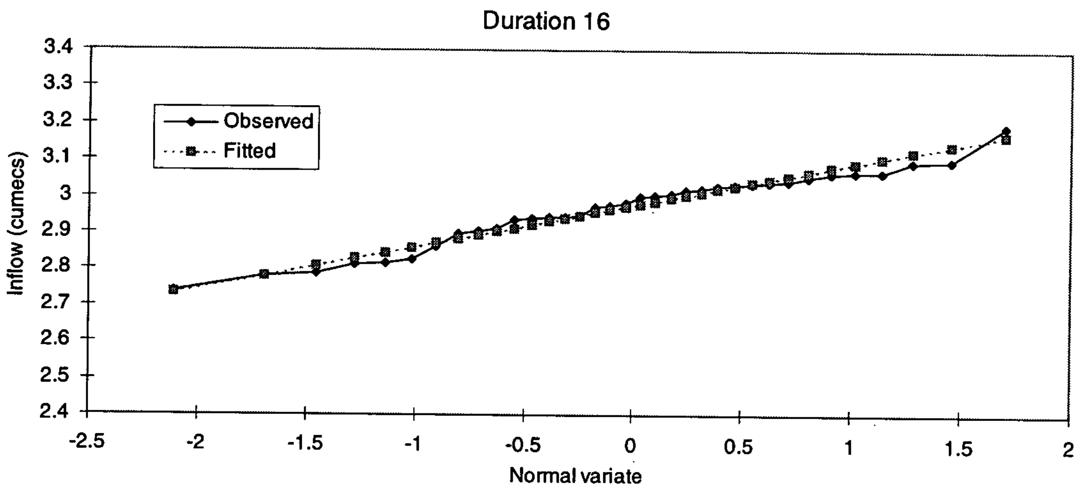

Duration 11
Inflow (cumecs)
Normal variate
Observed
Fitted
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

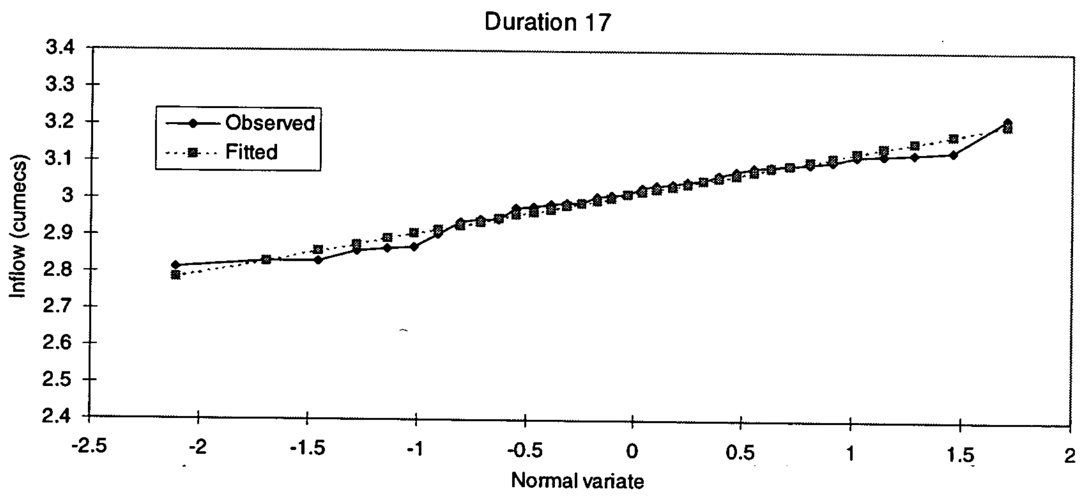

Duration 12
Observed
Fitted
Inflow (cumecs)
Normal variate
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Duration 13
Observed
Fitted
Inflow (cumecs)
Normal variate
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

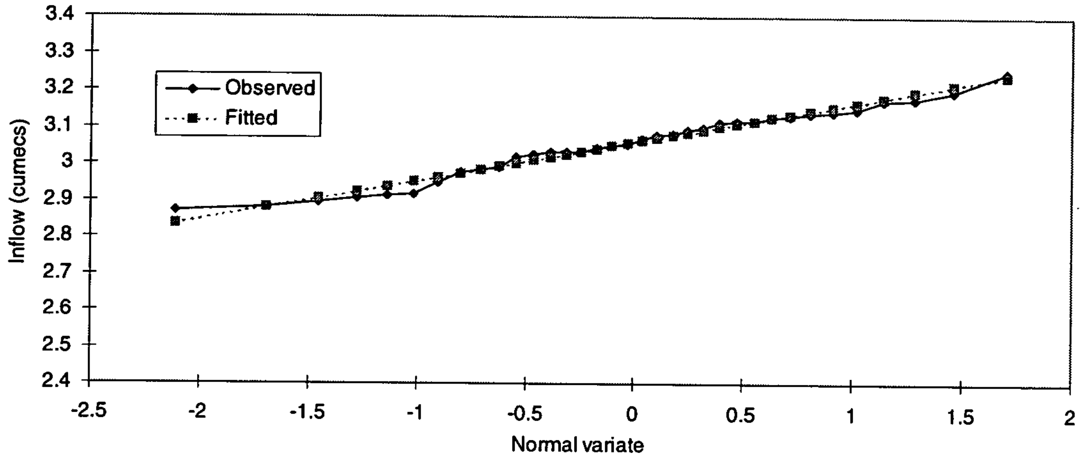

Duration 14
Observed
Fitted
Inflow (cumecs)
Normal variate
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Duration 15
Observed
Fitted
Inflow (cumecs)
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

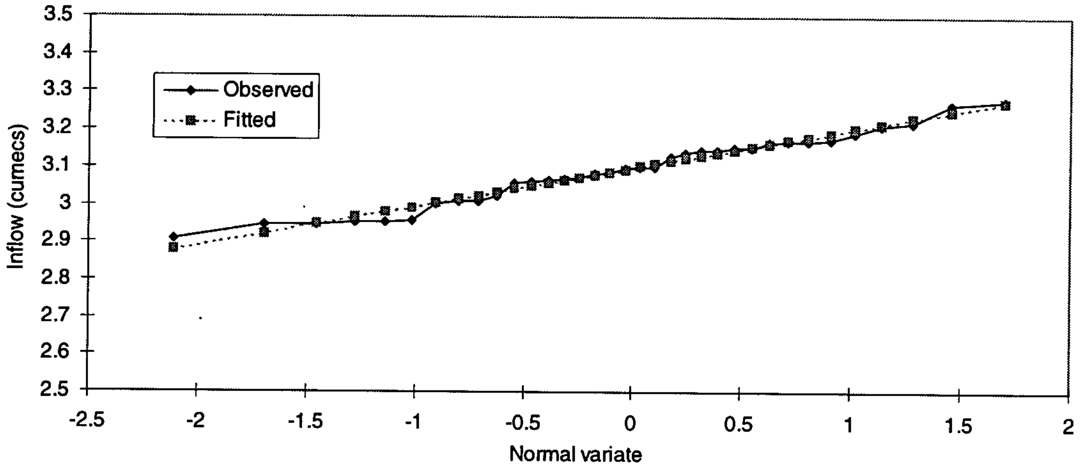

Duration 16
Observed
Fitted
Inflow (cumecs)
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate


Duration 17
Observed
Fitted
Inflow (cumecs)
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

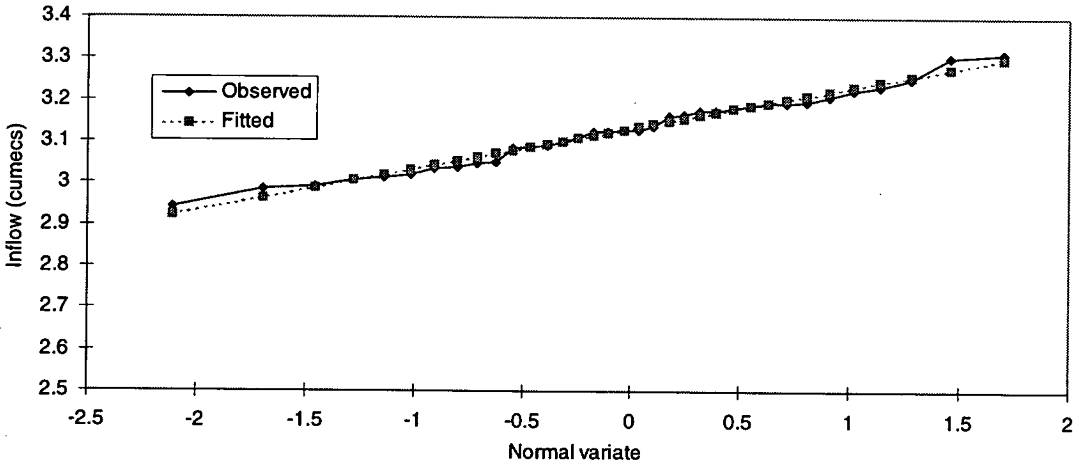
Duration 18

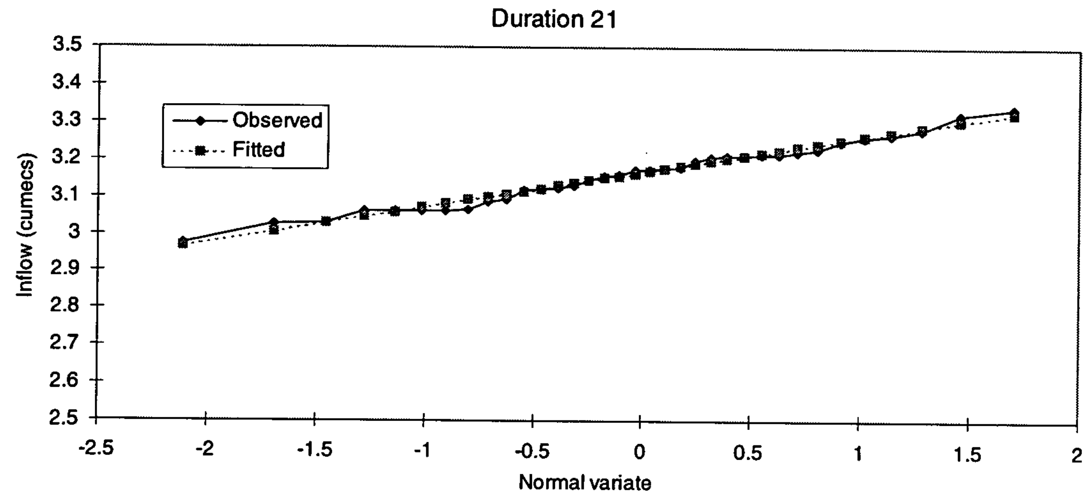


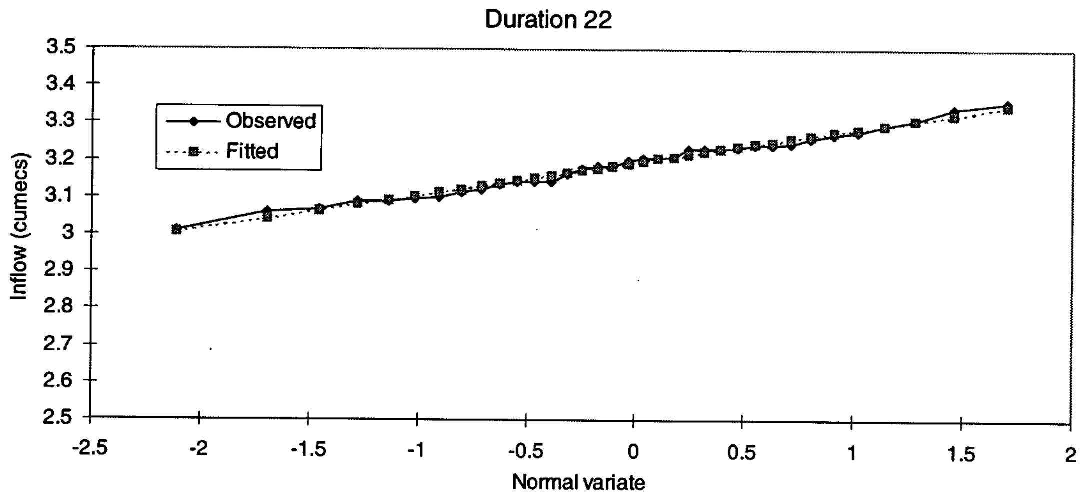
Duration 19

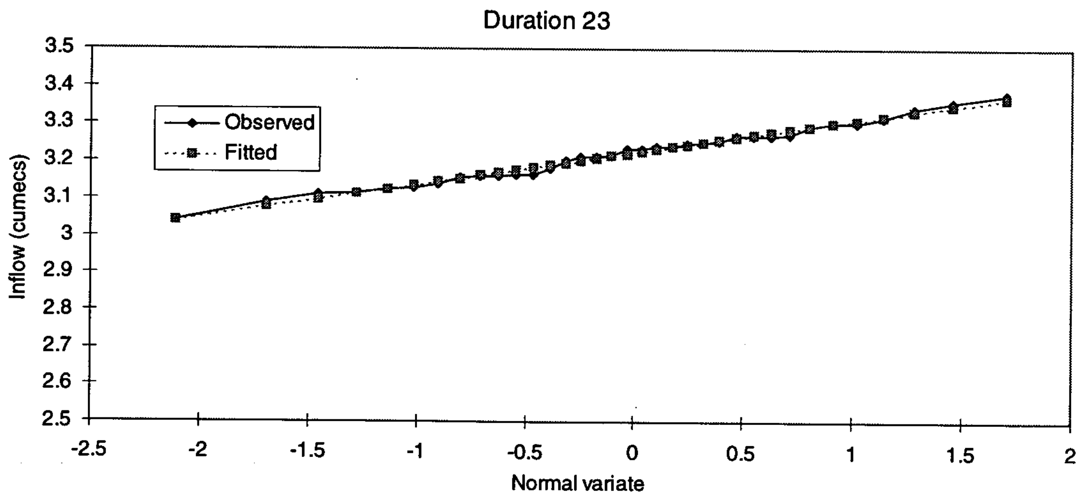


Duration 20

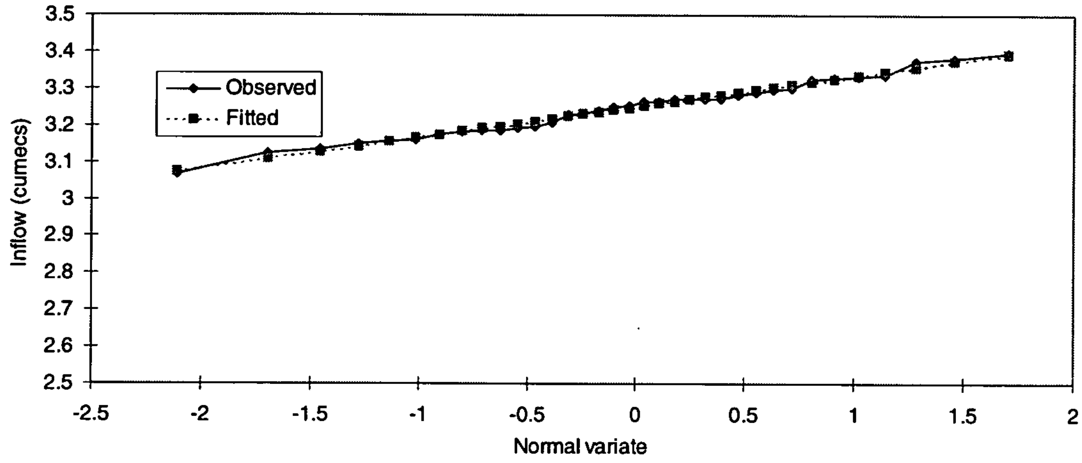



Duration 21
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
Inflow (cumecs)
Observed
Fitted
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

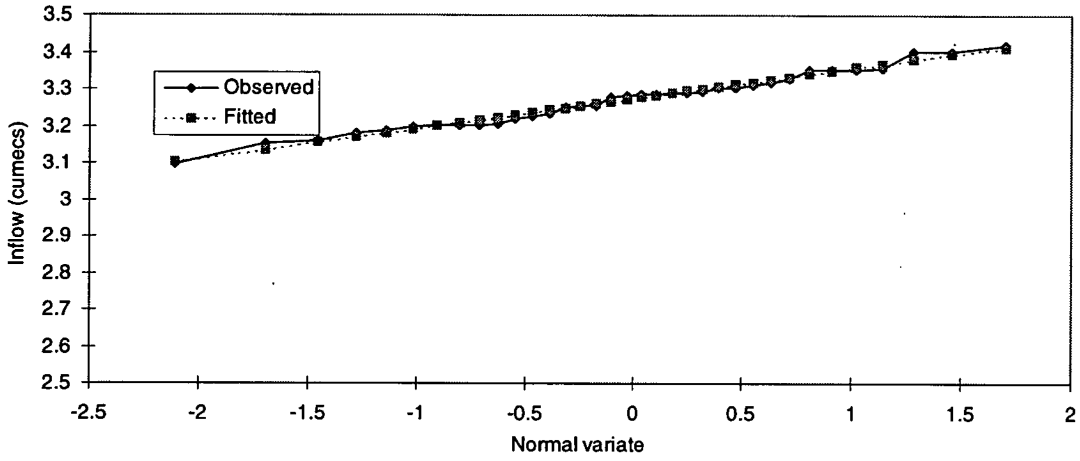

Duration 22
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
Inflow (cumecs)
Observed
Fitted
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate


Duration 23
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
Inflow (cumecs)
Observed
Fitted
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

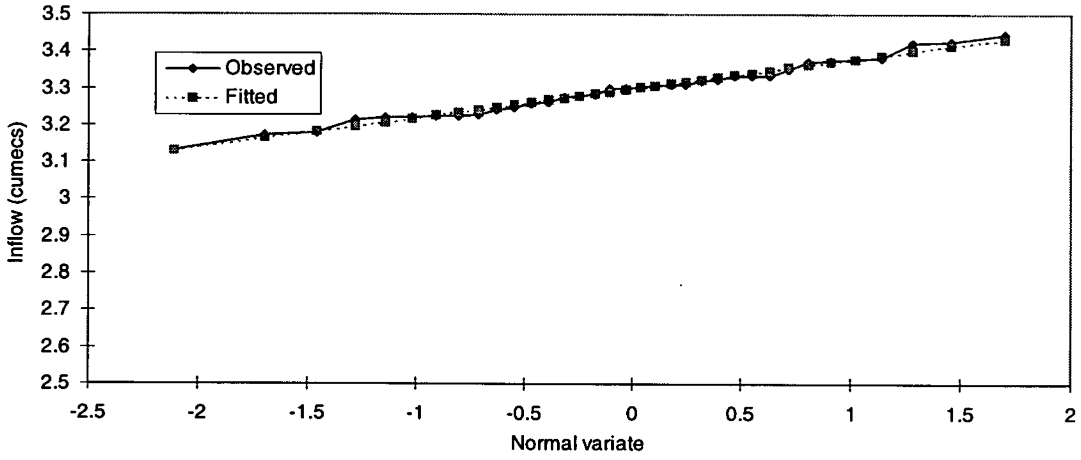
Duration 24

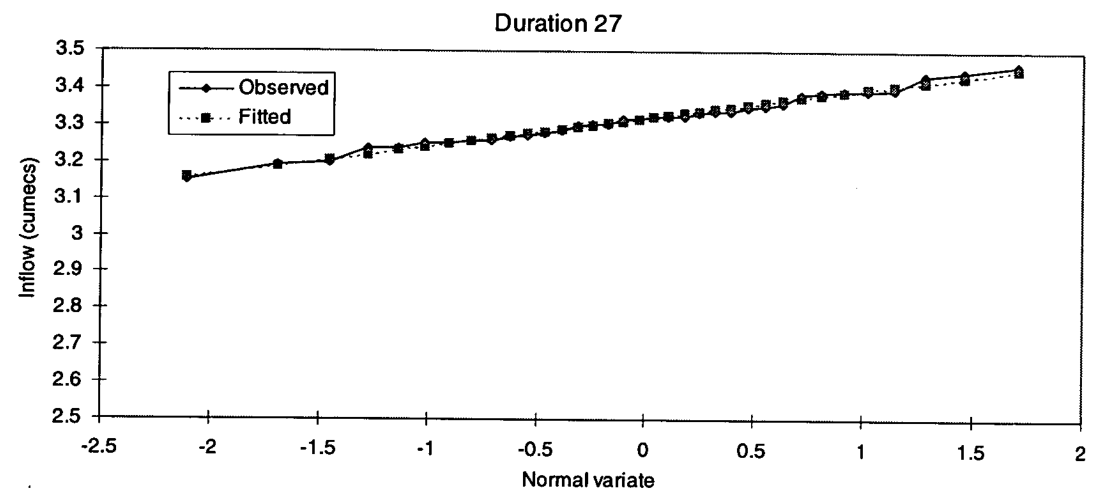


Duration 25

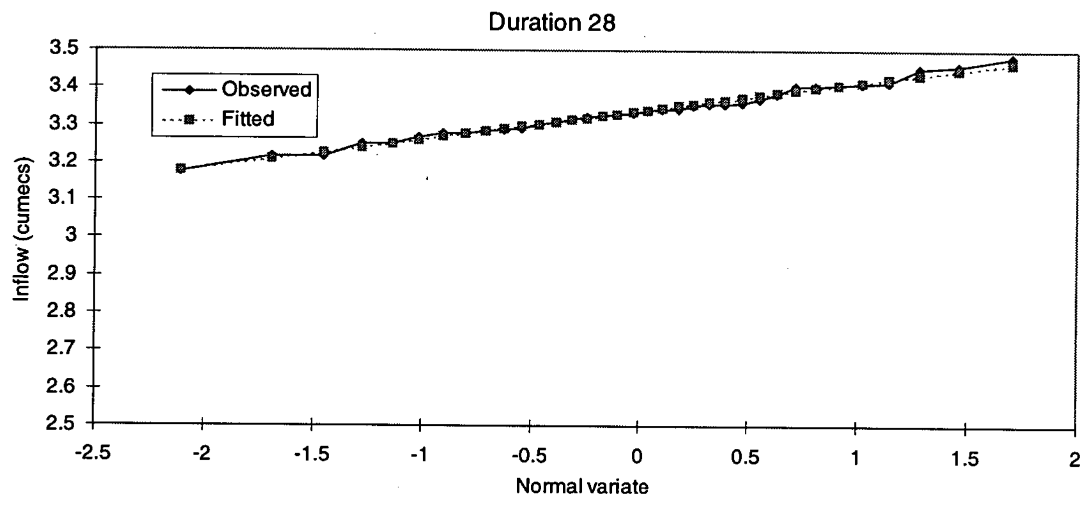


Duration 26

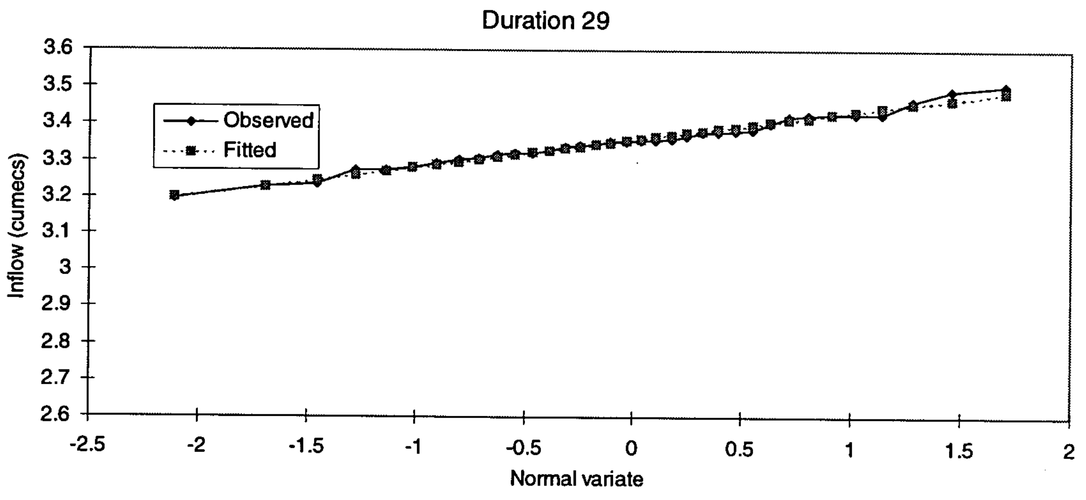




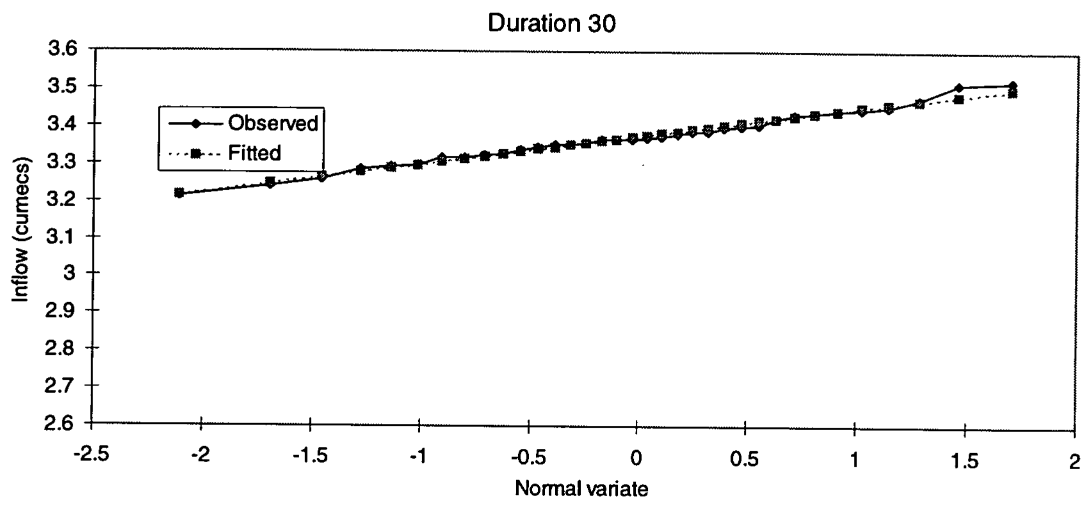
Duration 27
Observed
Fitted
Inflow (cumecs)
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

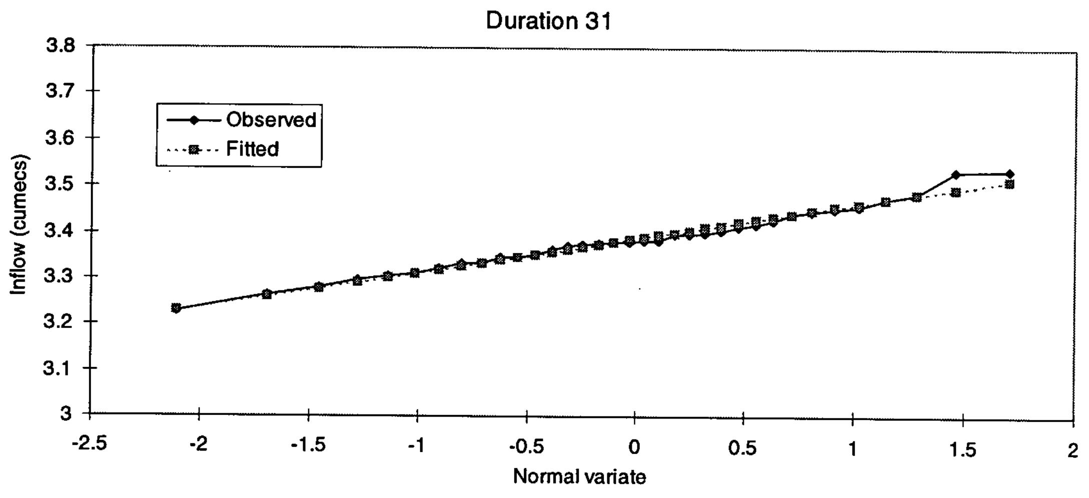


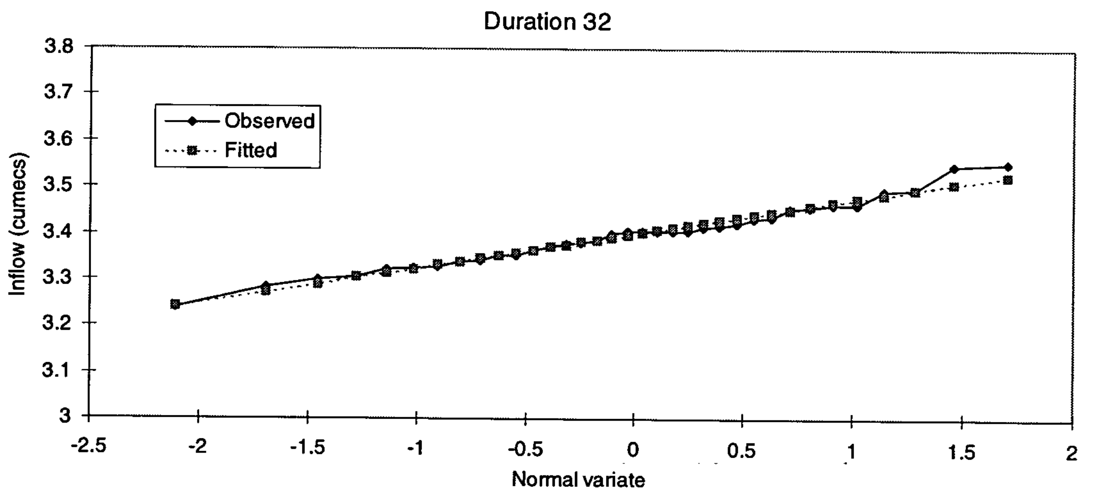
Duration 28
Observed
Fitted
Inflow (cumecs)
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

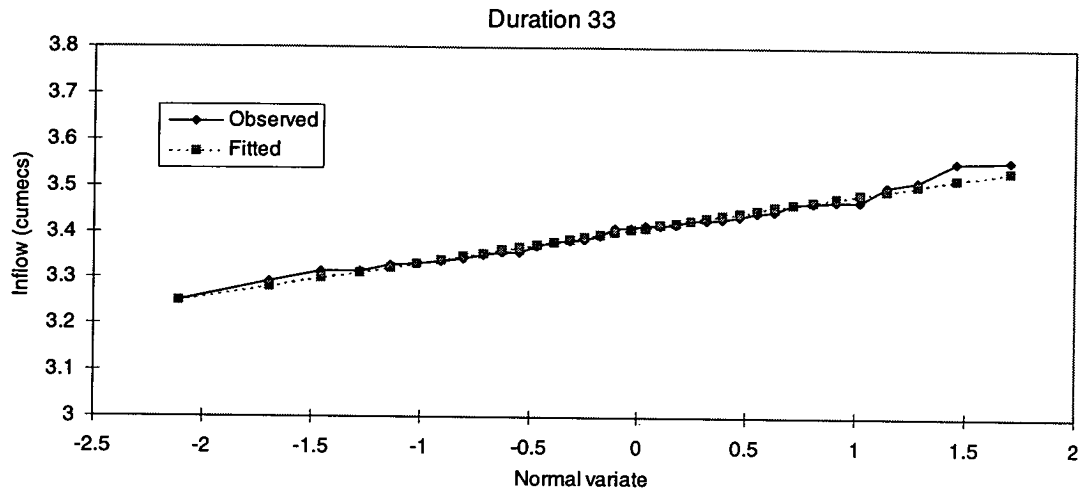


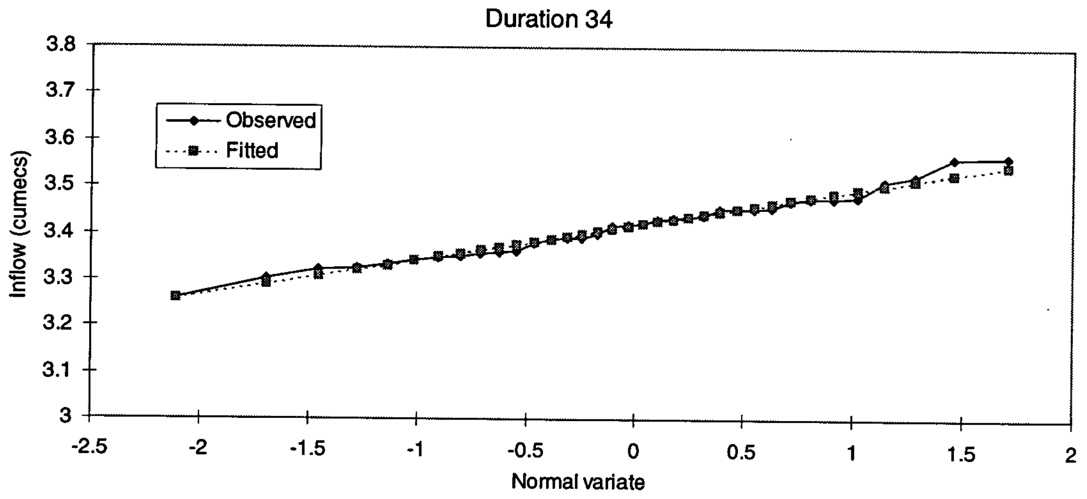
Duration 29
Observed
Fitted
Inflow (cumecs)
3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

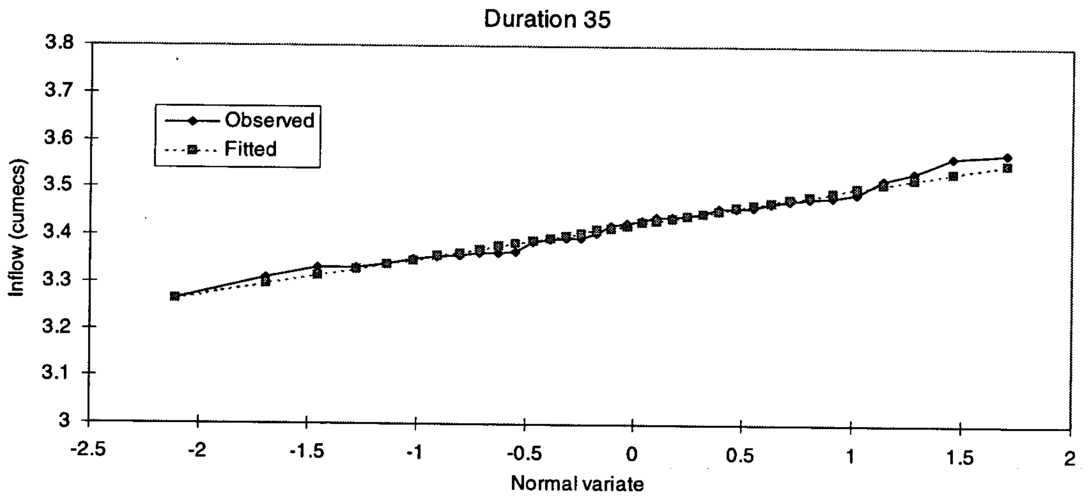

Duration 30
Inflow (cumecs)
Normal variate
Observed
Fitted
3.6
3.5
3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

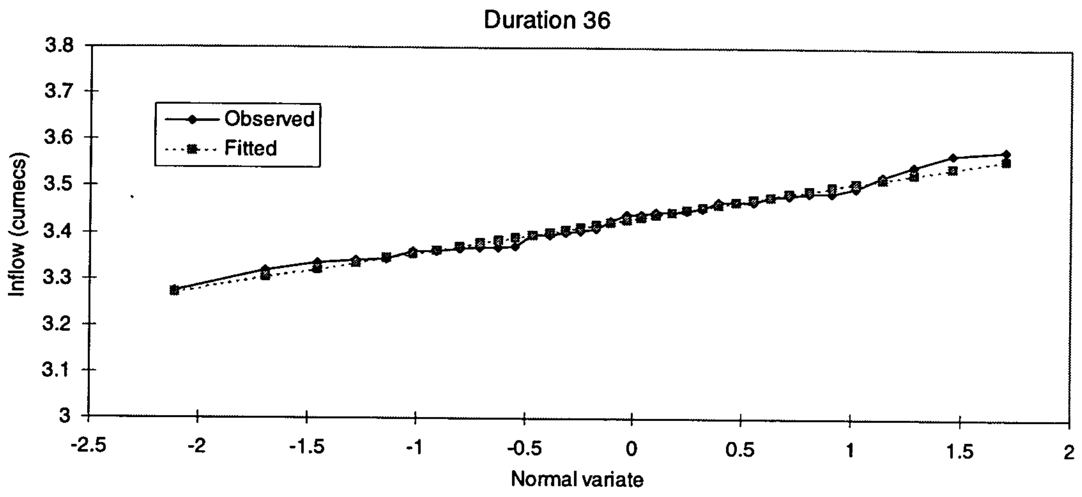

Duration 31
Inflow (cumecs)
Normal variate
Observed
Fitted
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Duration 32
Inflow (cumecs)
Normal variate
Observed
Fitted
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Duration 33
Observed
Fitted
Inflow (cumecs)
Normal variate
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Duration 34
Observed
Fitted
Inflow (cumecs)
Normal variate
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Duration 35
Observed
Fitted
Inflow (cumecs)
Normal variate
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Duration 36
Observed
Fitted
Inflow (cumecs)
3.8
3.7
3.6
3.5
3.4
3.3
3.2
3.1
3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Normal variate

B. Genetic Algorithm Computer Code

This appendix contains the source code for the program used to solve the four reservoir problem using a genetic algorithm. The code is developed using Microsoft Visual C++. It is written in a fairly generic form, and can be used to solve different multi reservoir optimisation problems with changes needed only to the files containing the data and the objective function of the problem.

The source code is contained in 5 files. An outline of these files is provided below.

File 1: MULRESGA.CPP

This file comprises a main program and a number of functions required to implement the GA procedure. The details of these functions are documented in the code itself. This file contains functions to implement different selection schemes such as proportional selection, tournament selection and rank based selection. There is a provision in the code to use one point, two point, or the uniform crossover scheme. The code also includes functions for implementing uniform mutation, non uniform mutation, and modified uniform mutation schemes.

File 2: 4R_HEAD.CPP

It is the header file where the population size, number of reservoirs in the system, and number of time steps in the optimisation horizon are defined by the user.

File 3:UTILITIES.CPP

Miscellaneous routines required by the main program are included in this file.

File 4: 4R_DATA.CPP

It contains the data for the four reservoir problem. The file could easily be modified to include the data pertaining to the problem to be solved with this code.

File 5: 4R_EVAL.CPP

It contains the objective function for the four reservoir problem. This file could be changed to incorporate the objective function of any other problem to be solved.

```
/***********************MULRESGA.CPP****************************/
/*        This is a real-valued genetic algorithm code which can be         */
/*        used to solve multi reservoir optimisation problems.              */
/*        The code is written in C++ and requires a set of five files       */
/*******************************************************************/
/*        FILE 4R_HEAD.CPP is the header file where the population          */
          size, number of reservoirs in the system, and number of time      */
          steps in the optimisation horizon are defined by the user         */
/*******************************************************************/
/*        FILE UTILITIES.CPP contains routines that are required by         */
/*        the main code                                                               */
/*******************************************************************/
/*        FILE 4R_DATA.CPP contains the data for the four reservoir         */
/*        problem. This file could easily be modified to include the        */
/*        data of  the problem to be solved with this code                  */
/*******************************************************************/
/*        FILE 4R_EVAL.CPP contains the objective function for the four  */
/*        reservoir problem.                                                */
/*******************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include<time.h>

#include "4R_HEAD.CPP"        /* the header file */
FILE *garun;                            /* an output file */
/*******************************************************************/
struct genotype    .
{
 int gene[NVARS];                                /* NVARS number of genes of a chromosome*/
                            /           * NVARS = num_reservoir * num_stages*/
 double fitness;            /* fitness of chromosomes        */
 int upper[NVARS];                      /* Upper limit of the genes*/
 int lower[NVARS];          /* Lower limit of the genes        */
 double rfitness;           /* relative fitness of a chromosome*/
 double cfitness;                       /* cumulative fitness of a chromosome*/
};

struct reschar                          /* struct defining reservoir characteristics */
{
 float storage[num_stages+1];  /* an array for storages*/
 float release[num_stages];    /* an array for releases*/
 float inflow[num_stages];     /* an array for inflows*/
 float stormin;                         /* minimum storage*/
 float stormax;                         /* initial storage*/
 float storinitial;                     /* final storage*/
 float storfinal;
};
float **ben;
#include "UTILITIES.CPP"
#include "4R_DATA.CPP"
#include "4R_EVAL.CPP"
/*******************************************************************/
/*        This function initializes a population of chromosomes within        */
/*        the lower and lower bounds of the variables of the problem.         */
/*        These bounds are specified through a data file which is        */
/*        pplied to this function as a second argument                        */
/*******************************************************************/
void initialize1(genotype population[], char fname[])
{
FILE *infile;
int i, j,lbound, ubound;

        if ((infile = fopen(fname,"r"))==NULL) {
                printf("Cannot open input file!\n");
                exit(1);
        }
```

```
/* initialize variables within the bounds */
            for (i = 0; i < NVARS; i++) {
                        fscanf(infile, "%d",&lbound);
                        fscanf(infile, "%d",&ubound);

                        for (j = 0; j < POPSIZE; j++){
                                    population[j].fitness = 0;
                                    population[j].rfitness = 0;
                                    population[j].cfitness = 0;
                                    population[j].lower[i] = lbound;
                                    population[j].upper[i]= ubound;
                                    population[j].gene[i] = randval(population[j].lower[i],
                        population[j].upper[i]);
                        }
            }
fclose(infile);
}
/*********************************************************************/
/*          This function is similar to initialize1 except that each GA              */
/*          run starts with a different number number seed                           */
/*********************************************************************/
void initialize2(genotype population[], char fname[])
{
            FILE *infile;
            int i, j,lbound, ubound;
            float rmax;

            if ((infile = fopen(fname,"r"))==NULL) {
                        printf("Cannot open input file!\n");
                        exit(1);
            }
            /* initialize variables within the bounds */
            /* set the initial random number seed */
            srand( (unsigned)time( NULL ) );
            rmax = RAND_MAX ;  printf(" rmax = %f (any key)\n", rmax) ;
            /* read upper and lower bounds from the file */
            for (i = 0; i < NVARS; i++) {
                        fscanf(infile, "%d",&lbound);
                        fscanf(infile, "%d",&ubound);

                        for (j = 0; j < POPSIZE; j++){
                                    population[j].fitness = 0;
                                    population[j].rfitness = 0;
                                    population[j].cfitness = 0;
                                    population[j].lower[i] = lbound;
                                    population[j].upper[i]= ubound;
                                    population[j].gene[i] = randval(population[j].lower[i],
                        population[j].upper[i]);
                        }
            }
fclose(infile);
}

/*********************************************************************/
/*          Keep_the_best function: This function keeps track of the           */
/*           best member of the population. Note that the last entry in        */
/*          the array Population holds a copy of the best individual           */
/*****************************************************************/
void keep_the_best(genotype population[])
{
            int i,mem;
            int cur_best;                   /* index of the best individual */
            cur_best = 0; /* stores the index of the best individual */
            double maxfitness = 0.0 ;

            for (mem = 0; mem < POPSIZE; mem++) {
                        if (population[mem].fitness > maxfitness) {
                                    maxfitness = population[mem].fitness;
                                    cur_best = mem;
```

```
                                population[POPSIZE].fitness = population[mem].fitness;
                        }
                }
/* once the best member in the population is found, copy the genes */
                for (i = 0; i < NVARS; i++)
                        population[POPSIZE].gene[i] = population[cur_best].gene[i];
}
/*******************************************************************/
/*      Elitist function: The best member of the previous generation        */
/*      is stored as the last in the array.                                 */
/*******************************************************************/
void elitist(genotype population[])
{
        int i;
        double best, worst;     /* best and worst fitness values */
        int best_mem, worst_mem; /* the best and the worst members */

        best = population[0].fitness;
        worst = population[0].fitness;

        for (i = 0; i < POPSIZE - 1; ++i) {
                if(population[i].fitness > population[i+1].fitness) {
      if (population[i].fitness >= best) {
          best = population[i].fitness;
          best_mem = i;
      }
      if (population[i+1].fitness <= worst) {
          worst = population[i+1].fitness;
          worst_mem = i + 1;
      }
    }
                else {
      if (population[i].fitness <= worst) {
          worst = population[i].fitness;
          worst_mem = i;
       }
      if (population[i+1].fitness >= best) {
          best = population[i+1].fitness;
          best_mem = i + 1;
      }
                }
   }
/*******************************************************************/
/*      if best individual from the new population is better than           */
/*      the best individual from the previous population, then              */
/*      copy the best from the new population; else replace the             */
/*      worst individual from the current population with the               */
/*      best one from the previous generation                               */
/*******************************************************************/
        if (best >= population[POPSIZE].fitness)  {
                for (i = 0; i < NVARS; i++)   {
                        population[POPSIZE].gene[i] = population[best_mem].gene[i];
                }
                population[POPSIZE].fitness = population[best_mem].fitness;
        }
        else  {
        for (i = 0; i < NVARS; i++) {
                population[worst_mem].gene[i] = population[POPSIZE].gene[i];
                }
                population[worst_mem].fitness = population[POPSIZE].fitness;
  }
}
/*******************************************************************/
/*      Selection function: Standard proportional selection                 */
/*******************************************************************/
void ProportionSelect(genotype population[],genotype newpopulation[])
{
        int mem, i, j;
        double sum = 0;
```

```
            double p,avg;

 /* find total fitness of the population */
            for (mem = 0; mem < POPSIZE; mem++) {
                      sum += population[mem].fitness;
            }
            avg = sum/(double)POPSIZE;

 /* calculate relative fitness */
            for (mem = 0; mem < POPSIZE; mem++) {
                      population[mem].rfitness =  population[mem].fitness/sum;
   }
            population[0].cfitness = population[0].rfitness;

 /* calculate cumulative fitness */
            for (mem = 1; mem < POPSIZE; mem++) {
     population[mem].cfitness =  population[mem-1].cfitness +
                population[mem].rfitness;
            }

 /* finally select chromosomes using cumulative fitness */
            for (i = 0; i < POPSIZE; i++) {
    p = rand()%1000/1000.0;
    if (p < population[0].cfitness)
        newpopulation[i] = population[0];
    else
        {
        for (j = 0; j < POPSIZE;j++)
            if (p >= population[j].cfitness &&
                   p<population[j+1].cfitness)
                newpopulation[i] = population[j+1];
        }
   }
/* once a new population is created, copy it back */
           for (i = 0; i < POPSIZE; i++) {
                     population[i] = newpopulation[i];
           }
}
/********************************************************************/
void BinaryTourSelect(genotype population[],genotype newpopulation[])
{
           int mem,ii,jj,i;
           /* find the best member from the group and select it */
           for (mem = 0; mem < POPSIZE; mem++) {
                     ii = randval(0,POPSIZE-1);
                     jj = randval(0,POPSIZE-1);
                     if(population[ii].fitness >= population[jj].fitness) newpopulation[mem]= population[ii];
                              else newpopulation[mem] =  population[jj] ;
           }
/* once a new population is created, copy it back */
           for (i = 0; i < POPSIZE; i++) {
                     population[i] = newpopulation[i];
           }
}
/***************************************************************/
int TournmSelection(genotype population[])
{
 int TS_K = 4;
 float TS_P = 0.25 ;
 float p;
 int i,j;

 typedef struct
 {
  int index;
  double fitness;
 } tournmType ;

 tournmType sel[POPSIZE];
```

```
  tournmType aux;

   /* Selects K individuals */

            for(i=0;i<TS_K;i++) {
                     sel[i].index=randval(0,POPSIZE-1);
                     sel[i].fitness=population[sel[i].index].fitness;
   }

  /* Sorts the fitness/ Brings best to the top */
  for(i=0;i<TS_K-1;i++) {

             for (j=i+1;j<TS_K;j++) {
                     if (sel[i].fitness<=sel[j].fitness) {
                              aux=sel[i];
                              sel[i]=sel[j];
                              sel[j]=aux;
                     }
             }
  }
          p = rand()%1000/1000.0F;
          if (p<TS_P) sel[0].index = sel[1].index ;
          return(sel[0].index);
}
/*********************************************************************/
void ModTourSelect(genotype population[],genotype newpopulation[])
{
          int i, ii;

          for (i = 0; i < POPSIZE; i++) {
                    ii = TournmSelection(population) ;
                    newpopulation[i] = population[ii];
          }

          /* copy back the poulation */
          for (i = 0; i < POPSIZE; i++) {
                    population[i] = newpopulation[i];

          }
}
/*********************************************************************/
/*          MODIFIED SELECTION SCHEME                                       */
/******************************************************************/
void mod_select(genotype population[],genotype newpopulation[])
{
int mem, i, j;
double sum = 0;
double p;

/* find total fitness of the population */
          for (mem = 0; mem < POPSIZE; mem++) {
                    population[mem].fitness = 1- mem/(POPSIZE-1);
                    sum += population[mem].fitness;
          }
/* calculate relative fitness */
          for (mem = 0; mem < POPSIZE; mem++) {
                    population[mem].rfitness =  population[mem].fitness/sum;
          }
          population[0].cfitness = population[0].rfitness;

/* calculate cumulative fitness */
          for (mem = 1; mem < POPSIZE; mem++) {
   population[mem].cfitness =  population[mem-1].cfitness +
              population[mem].rfitness;
          }
/* finally select survivors using cumulative fitness. */
          for (i = 0; i < POPSIZE; i++) {
   p = rand()%1000/1000.0;
   if (p < population[0].cfitness)
```

```
            newpopulation[i] = population[0];
        else
            {
            for (j = 0; j < POPSIZE;j++)
                if (p >= population[j].cfitness &&
                        p<population[j+1].cfitness)
                    newpopulation[i] = population[j+1];
            }
    }
/* once a new population is created, copy it back */
            for (i = 0; i < POPSIZE; i++) {
                    population[i] = newpopulation[i];
            }
}
/***************************************************************************/
void Xover(int one, int two,genotype population[])
{
int i;
int point; /* crossover point */

/* select crossover point */
            point = num_reservoir * randval(1,num_stages -1) ;

  for (i = point; i < NVARS; i++)
      swap(&population[one].gene[i], &population[two].gene[i]);
}
/***************************************************************************/
/*          Crossover: performs 2 point crossover of the two selected parents.  */
/***************************************************************************/
void TPXover(int one, int two,genotype population[])
{
int i;
int point1,point2; /* crossover point */

/* select crossover point */

      point1 = num_reservoir * randval(0,num_stages -1) ;
      point2 = num_reservoir * randval(0,num_stages -1) ;

                    if(point2>point1) swap(&point1, &point2);

                    for (i = point1; i < point2; i++)  {
            swap(&population[one].gene[i], &population[two].gene[i]);
                    }
}
/*********************************************************************/
void MUNXover(int one, int two,genotype population[])
{
          int i,ii,crossindex,kk;
          int blockLength ;
          kk=0;

          blockLength = num_reservoir ;

  for (ii=0; ii<num_stages; ii++) {
          crossindex = rand()&01;
                    for (i = kk; i < kk + blockLength; i++)  {
                    if (crossindex ==0) swap(&population[one].gene[i], &population[two].gene[i]);
                    }
                    kk=kk+blockLength;
          }

}
/***************************************************************************/
/* Performs uniform crossover                                               */
/***************************************************************************/
void UNXover(int one, int two,genotype population[])
{
          int i,rr,crossindex,kk;
```

```
            int blockLength ;
            kk=0;
            blockLength = num_stages ;
    for (rr=0; rr<num_reservoir; rr++) {
            crossindex = rand()&01;
                    for (i = kk; i < kk + blockLength; i++)  {
                    if (crossindex ==0) swap(&population[one].gene[i], &population[two].gene[i]);
                    }
                    kk=kk+blockLength;
            }
}
/*********************************************************************/
/*          Crossover selection: selects two chromosomes that take part          */
/*          in  the crossover. Implements one point crossover                    */
/*********************************************************************/
void OnePointCrossover(genotype population[],float PXOVER)
{
int mem, one;
int first  =  0; /* count of the number of members chosen */
double x;

            for (mem = 0; mem < POPSIZE; ++mem) {
    x = rand()%1000/1000.0;
                    if (x < PXOVER ) {
        ++first;
        if (first % 2 == 0)
          Xover(one, mem,population);
        else
            one = mem;
                    }
    }
}
/*********************************************************************/
/*          Two point crossover scheme                                           */
/*********************************************************************/
void TwoPointCrossover(genotype population[],float PXOVER)
{
int mem, one;
int first  =  0;            /* count of the number of members chosen */
double x;

            for (mem = 0; mem < POPSIZE; ++mem) {
    x = rand()%1000/1000.0;
                    if (x < PXOVER ) {
        ++first;
        if (first % 2 == 0)
          TPXover(one, mem,population);
        else
            one = mem;
                    }
    }
}
/*********************************************************************/
/*          Uniform Crossover : selects two parents that take part in            */
/*          the crossover. Implements  uniform crossover               */
/*********************************************************************/
void UniformCrossover(genotype population[],float PXOVER)
{
int mem, one;
int first  =  0; /* count of the number of members chosen */
double x;

            for (mem = 0; mem < POPSIZE; ++mem) {
    x = rand()%1000/1000.0;
                    if (x < PXOVER ) {
        ++first;
        if (first % 2 == 0)
          MUNXover(one, mem,population);
        else
```

```
            one = mem;
                        }
    }
 }
/************************************************************************/
/*          Modified Uniform Mutation. A gene selected for                 */
/*          mutation is modified by a predefined small amount which        */
/*          may be positive or negative with equal probability             */
/************************************************************************/
void ModUniMutate(genotype population[],float PMUTATION)
{
int i, j;
int lbound, hbound;
double x;
int mutant,change;
change = 1;

            for (i = 0; i < POPSIZE; i++) {
            for (j = 0; j < NVARS; j++) {
        x = rand()%1000/1000.0;
                                    if (x < PMUTATION ) {
          /* find the bounds on the variable to be mutated */
          lbound = population[i].lower[j];
          hbound = population[i].upper[j];
                      mutant = rand()&01 ;
           if (mutant==1) {
           population[i].gene[j] = population[i].gene[j] + change;
           if (population[i].gene[j] > hbound) population[i].gene[j] = population[i].gene[j] -
                                              2*change ;
           }
           else {
           population[i].gene[j] = population[i].gene[j] - change;
           if (population[i].gene[j] < lbound) population[i].gene[j] = population[i].gene[j] +
                                              2*change ;
           }
        }
           }
    }
}   /* end of routine */
/************************************************************************/
/*          UNIFORM MUTATION: Random uniform mutation. A gene              */
/*          selected for mutation is replaced by a random value between lower */
/*          and upper bounds of this gene                                  */
/************************************************************************/
/************************************************************************/
void UniMutate(genotype population[],float PMUTATION)
{
int i, j;
int lbound, hbound;
double x;

           for (i = 0; i < POPSIZE; i++) {
                      for (j = 0; j < NVARS; j++) {
                                 x = rand()%1000/1000.0;
                                 if (x < PMUTATION ) {
                                 /* find the bounds on the variable to be mutated */
                      lbound = population[i].lower[j];
                       hbound = population[i].upper[j];
                      population[i].gene[j] = randval(lbound,hbound);
                                 }
                      }
           }
}
/************************************************************************/
/* Report function: Reports progress of the simulation.               */
/************************************************************************/
void report(genotype population[],int generation)
{
int i;
```

```
double best_val;                /* best population fitness */
double avg;                     /* avg population fitness */
double sum_square;              /* sum of square for std. calc */
double square_sum;              /* square of sum for std. calc */
double sum;                     /* total population fitness */
sum = 0.0;
sum_square = 0.0;

        for (i = 0; i < POPSIZE; i++) {
                sum += population[i].fitness;
                sum_square += population[i].fitness * population[i].fitness;
        }

        avg = sum/(double)POPSIZE;
        square_sum = avg * avg * POPSIZE;
/*  stddev = sqrt((sum_square - square_sum)/(POPSIZE - 1));  */
        best_val = population[POPSIZE].fitness;
        fprintf(garun, "%4d    %5.3f  %5.3f  %5.2f\n", generation,best_val, avg);
        printf("%4d  %4.3f  %4.3f\n", generation,best_val,avg);
}
/*********************************************************************/
/*      Main function: The GA procedure is repeated for the required     */
/*      number of generations                                                      */
/*********************************************************************/
void main(void)
{
int ii,generation, dummy;
float Increase,oldfitness,newfitness;

char title[80];
char garunfile[20],benfuncfile[20],
        datafile[20], resultfile[20];
FILE *f1;

        struct genotype population[POPSIZE+1];          /* population */
        struct genotype newpopulation[POPSIZE+1];       /* new population; */
        struct reschar res[num_reservoir];
        int MAXGENS;                                        /* Maximum number of generations*/
        float PXOVER;                                       /* probability of crossover */
        float PMUTATION;                                /* probability of mutation */

        /* Open the list file */
        if(!(f1 = fopen("4r_ga12.lst", "r"))) {
        printf(" Can't open 4r_ga**.lst on f1/n") ;
        exit(1) ;
  }
        fgets(title, 80, f1);
        fscanf(f1,"%d\n",&dummy);
        if(num_stages!=dummy) {
                printf("check num_stages\n");
                exit(1);
        }
        puts(title);
        fgets(garunfile,20,f1);
        fgets(benfuncfile,20,f1);
        fgets(datafile,20,f1);
        fgets(resultfile,20,f1);

        for (ii=0; ii<20; ii++) {
                if (garunfile[ii] =='\n') garunfile[ii]='\0';
                if (benfuncfile[ii] =='\n') benfuncfile[ii]='\0';
                if (datafile[ii] =='\n') datafile[ii]='\0';
                if (resultfile[ii] =='\n') resultfile[ii]='\0';
        }
        (void)fclose(f1) ;
        /* Open an output file */
                if ((garun = fopen(garunfile,"w"))==NULL) {
                exit(1);
                }
```

```
            printf("Enter the crossover probability:  ");
            scanf("%f", &PXOVER) ;
            printf("Enter the mutation  probability:  ");
            scanf("%f", &PMUTATION) ;
            printf("Enter the  number of generation:  ");
            scanf("%d", &MAXGENS) ;
/************************************************************/
            ben = matrix(0, num_stages - 1, 0, 4) ;
            /* read the benefit functions from the benfuncfile */
            readdata(res,benfuncfile);
            /* start the generation loop */
            generation = 0;
            initialize1(population,datafile);           /* initialise the population randomly */
            /* initialize2(population,datafile);                */
            evaluate(population,res);                   /* evaluate the fitness of schromosomes */
            keep_the_best( population);
            Increase = 10.0F;
            oldfitness = 0.0F;
            while((Increase >= 0.1F) && (generation<MAXGENS))        {
                        generation++;
                        report(population,generation);
/*                      rank_select(); */
                        ModTourSelect(population,newpopulation);
/*                      ProportionSelect(); */
                        UniformCrossover(population,PXOVER);    /* Uniform */
/*                      OnePointCrossover(); */
/*                      TwoPointCrossover();*/
                        ModUniMutate(population,PMUTATION);
/*                      UniMutate(population,PMUTATION);        */
                        evaluate(population,res);
                        elitist(population);
                        if((generation)%100==0) {
                        newfitness = population[POPSIZE].fitness;
                        printf("oldfitness = %3.3f\n", oldfitness);
                        printf("newfitness = %3.3f\n", newfitness);
                        Increase =  newfitness - oldfitness;
                        printf("Increase = %3.3f\n", Increase);
                        oldfitness = newfitness;
                        }
            }
            /* print the results */
            result(population,res,MAXGENS,PXOVER,PMUTATION,resultfile);
            fclose(garun);
            printf("Run Complete\n");
}/*end of main */
```

```
/*********************************************************************/
/* File 4R_HEAD.CPP: Header file */
/*********************************************************************/
#define POPSIZE 100
#define num_reservoir 4
#define num_stages 12
#define NVARS num_reservoir * num_stages
```

```
/************************************************************************/
/* FILE 4R_DATA.CPP: Reads data from the supplied files                 */
/************************************************************************/
void readdata(reschar res[], char fname[])
{
        FILE *f3;
        int ii,jj,rr;
        int stage_index,reservoir_index ;
        char title[80] ;

        /* Open the benefit functions data file */
        if(!(f3 = fopen(fname, "r"))) {
        nrerror(" Can't open ben_func.dat on f3/n") ;
        exit(1) ;
  }

  fgets(title, 80, f3) ;
  for(stage_index = 0; stage_index < num_stages; stage_index++) {
        ii = stage_index ;
                for(jj = 0; jj < 5; jj++) {
                        fscanf(f3,"%f",&ben[ii][jj]) ;
                }
  }

        (void)fclose(f3) ;
/************************************************************************/
/*          assigns inflows to reservoirl and reservoir 2                */
/************************************************************************/  for(stage_index = 0; stage_index <
num_stages; stage_index++) {
                ii = stage_index ;
                res[0].inflow[ii] = 2.0F;
                res[1].inflow[ii] = 3.0F;
        }
/*******************************************************************/
/* Assigning max. and min. storages to different reservoirs         */
/*******************************************************************/
        for(reservoir_index = 0;  reservoir_index < num_reservoir;
                reservoir_index++) {
                rr=reservoir_index ;
                res[rr].stormin =  0.0 ;
                res[rr].stormax = 10.0 ;
                res[rr].storfinal = 5.0 ;
                res[rr].storinitial = 5.0 ;
                if (rr==3) res[rr].stormax = 15.0 ;
                if (rr==3) res[rr].storfinal = 7.0 ;
        }

        for(reservoir_index = 0;  reservoir_index < num_reservoir; reservoir_index++) {
                rr=reservoir_index ;
                res[rr].storage[0] = res[rr].storinitial ;
        }
/************************************************************************/
}
```

```
/*********************************************************************/
/*          4R_EVAL.CPP: contains evaluation function of the problem          */
/*********************************************************************/
/*          Evaluation function: This computes the fitness of each          */
/*          chromosome in the population                                    */
/*********************************************************************/
void evaluate(genotype population[],reschar res[])
{
        int mem;
        int i;
        int ii,rr ;
        float pen,sum_benefit, sumpenalty,penalty,pfactor;
        double nfactor = 1.0;
        float *benefit ;
        float minfactor = 2.0;
        float maxfactor =  1.0 ;
/*********************************************************************/
        benefit = vector(0, num_stages-1) ;
        /* start the population loop */
        for (mem = 0; mem < POPSIZE; mem++) {
                pen =0;
                for (ii = 0; ii < num_stages; ii++) {
                        for (rr = 0; rr < num_reservoir; rr++) {
                                i = ii * num_reservoir + rr ;
                                res[rr].release[ii] = (float)population[mem].gene[i];
                        }
                }

                for (ii = 0; ii < num_stages; ii++) {
                        res[0].storage[ii+1] = res[0].storage[ii] + res[0].inflow[ii] -
                        res[0].release[ii] ;
                                if(res[0].storage[ii+1] <= res[0].stormin)
                                        pen = pen+(res[0].storage[ii+1] - res[0].stormin)
                                   *(res[0].storage[ii+1] - res[0].stormin) * minfactor;
                                if(res[0].storage[ii+1] >= res[0].stormax)
                                        pen = pen+(res[0].storage[ii+1]-res[0].stormax)
                                        *(res[0].storage[ii+1]-res[0].stormax)*maxfactor;

                        res[1].storage[ii+1] = res[1].storage[ii] + res[1].inflow[ii] - r
                        es[1].release[ii] ;
                                if(res[1].storage[ii+1] <= res[1].stormin)
                                        pen = pen +(res[1].storage[ii+1]-res[1].stormin)*
                                        (res[1].storage[ii+1]-res[1].stormin)*minfactor;
                                if (res[1].storage[ii+1] >= res[1].stormax )
                                pen = pen +(res[1].storage[ii+1] - res[1].stormax)
                                *(res[1].storage[ii+1] - res[1].stormax)*maxfactor ;

                        res[2].inflow[ii] = (res[1].release[ii]) ;

                        res[2].storage[ii+1] = res[2].storage[ii] + res[2].inflow[ii] -
                        res[2].release[ii] ;
                                if(res[2].storage[ii+1] <= res[2].stormin)
                                pen = pen +(res[2].storage[ii+1]-res[2].stormin)
                                        *(res[2].storage[ii+1]-res[2].stormin)*minfactor;
                                if( res[2].storage[ii+1] >= res[2].stormax )
                                pen = pen +(res[2].storage[ii+1]- res[2].stormax)
                                *(res[2].storage[ii+1]- res[2].stormax)*maxfactor;

                        res[3].inflow[ii]=(res[0].release[ii] + res[2].release[ii]) ;

                        res[3].storage[ii+1] = res[3].storage[ii] + res[3].inflow[ii] -
                                                res[3].release[ii] ;
                                if(res[3].storage[ii+1] <= res[3].stormin)
                                        pen = pen +(res[3].storage[ii+1]-res[3].stormin)
                                        *(res[3].storage[ii+1]-res[3].stormin)*minfactor;
                                if(res[3].storage[ii+1] >= res[3].stormax )
                                        pen = pen +(res[3].storage[ii+1]-res[3].stormax)
                                        *(res[3].storage[ii+1]-res[3].stormax)*maxfactor;
                        benefit[ii] = ben[ii][0]* (res[0].release[ii])+ben[ii][1]* (res[1].release[ii])
                        + ben[ii][2]* (res[2].release[ii])+ ben[ii][3]* (res[3].release[ii])+
                        ben[ii][4]* (res[3].release[ii]) ;
                }
/*********************************************************************/
                sum_benefit = findsum(benefit, num_stages);
                pfactor = 10.0;
                sumpenalty = 0.0F;
                for(rr = 0; rr < num_reservoir; rr++) {
                        if(res[rr].storage[num_stages]<res[rr].storfinal) {
                                penalty = (res[rr].storage[num_stages]-res[rr].storfinal)*
                                (res[rr].storage[num_stages]-res[rr].storfinal)*(-1.0F * pfactor) ;
                        }
                        else penalty = 0.0 ;
                                sumpenalty+=penalty;
                }
        population[mem].fitness = (sum_benefit -pen + sumpenalty)/nfactor;
                if(population[mem].fitness<0.0) population[mem].fitness = 0.0 ;
/*********************************************************************/
        } /* population loop */
} /* end of function "evaluate " */
/*********************************************************************/
/*********************************************************************/
void result(genotype population[], reschar res[],int MAXGENS,
                        float PXOVER,float PMUTATION,char fname[])
{
int mem;
int i;
int ii,rr ;
float sum_benefit, penalty1,penalty2,penalty3,penalty4,pfactor;
float * benefit;
FILE *f4;

benefit = vector(0, num_stages-1) ;
/*********************************************************************/
  /* open an outout file */
        if(!(f4 = fopen(fname, "w"))) {
                nrerror(" Can't open output file 4r_mga.txt on f1/n") ;
                exit(1) ;
        }
        fprintf(f4,"POPSIZE : %d\n",POPSIZE);
        fprintf(f4,"MAXGENS : %4d\n",MAXGENS);
        fprintf(f4,"PXOVER  : %1.3f\n",PXOVER);
        fprintf(f4,"PMUTATE : %1.3f\n",PMUTATION);
/*********************************************************************/
    mem = POPSIZE ;
for (ii = 0; ii < num_stages; ii++) {
        for (rr = 0; rr < num_reservoir; rr++) {
                i = ii*num_reservoir + rr ;
                res[rr].release[ii] = (float)population[mem].gene[i];
                res[0].storage[ii+1] = res[0].storage[ii] + res[0].inflow[ii] - res[0].release[ii] ;
                if(rr>=1) {
                        res[1].storage[ii+1] = res[1].storage[ii] + res[1].inflow[ii] -
                                                        res[1].release[ii] ;
                        res[2].inflow[ii] = res[1].release[ii] ;
                        if(rr>=2) {
                        res[2].storage[ii+1] = res[2].storage[ii] + res[2].inflow[ii] -
                                                        res[2].release[ii];

                        res[3].inflow[ii]=(res[0].release[ii] + res[2].release[ii]) ;
                        if(rr>=3) {
                      res[3].storage[ii+1] = res[3].storage[ii] + res[3].inflow[ii] -
                                                                res[3].release[ii] ;
                benefit[ii] = ben[ii][0]* (res[0].release[ii])+ben[ii][1]* (res[1].release[ii])
                + ben[ii][2]* (res[2].release[ii])+ ben[ii][3]* (res[3].release[ii])+ben[ii][4]*
                                                (res[3].release[ii]) ;
                }
                }
                }
                        } /* reservoir loop */
```

```
                    } /* stage loop */
/***********************************************************************/
                    sum_benefit = findsum(benefit,num_stages);
                    pfactor = 10.0 ;
                    if(res[0].storage[num_stages]<5.0 ) {
                    penalty1 = (res[0].storage[num_stages]-5.0F)*(res[0].storage[num_stages]-5.0F)*(-
                                                            1.0F * pfactor) ;
                    }
                    else penalty1 = 0.0 ;
                    if(res[1].storage[num_stages]<5.0 ) {
                    penalty2= (res[1].storage[num_stages]-5.0F)*(res[1].storage[num_stages]-5.0F)*(-
                                                            1.0F * pfactor) ;
                    }
                    else penalty2 = 0.0 ;
                    if(res[2].storage[num_stages]<5.0 ) {
                    penalty3 = (res[2].storage[num_stages]-5.0F)*(res[2].storage[num_stages]-5.0F)*(-
                                                            1.0F * pfactor) ;
                    }
                    else penalty3= 0.0 ;

                    if(res[3].storage[num_stages]<7.0 ) {
                    penalty4 = (res[3].storage[num_stages]-7.0F)*(res[3].storage[num_stages]-7.0F)*(-
                                                            1.0F * pfactor) ;
                    }
                    else penalty4 = 0.0 ;

                    population[mem].fitness = sum_benefit + penalty1 + penalty2 + penalty3 +
                    penalty4;
                    if(population[mem].fitness<0.0) population[mem].fitness = 0.0 ;
                    /*********************************************/
          printf("FITNESS = %3.2f\n", population[mem].fitness);
          fprintf(f4," Reservoir1           Reservoir2         Reservoir3        Reservoir4\n" );
          fprintf(f4," STOR  INFL  RELE  STOR  INFL  RELE  STOR  INFL  RELE  STOR
          INFL  RELE \n");
          for (ii = 0; ii < num_stages; ii++) {
                    for (rr = 0; rr < num_reservoir; rr++) {
                    fprintf(f4,"%5.2f  %5.2f  %6.2f  ",
                              res[rr].storage[ii],res[rr].inflow[ii],res[rr].release[ii]);
                    }
                    fprintf(f4,"\n") ;
          }
                    fprintf(f4,"Maximum Benefit =%3.2f\n",population[mem].fitness);
                    (void) fclose(f4);
} /* end of routine "result " */
/***********************************************************************/
```

```
/***********************************************************************/
/*          UTILITIES.CPP: contains miscelleneous routines              */
/***********************************************************************/
int **imatrix(int nrl, int nrh, int ncl, int nch) ;
float **matrix(int nrl, int nrh, int ncl, int nch) ;
int *ivector(int nl, int nh) ;
float *vector(int nl, int nh) ;
void nrerror(char *messg); /*function prototype for error routine*/
void get_f(char name[]);
int randval(int, int);
float findsum(float [],int);  /*function prototype for summing n elements*/
/***********************************************************************/
/*          Swap: Used for swapping 2 variables                         */
/***********************************************************************/
void swap(int *x, int *y)
{
int temp;
temp = *x;
*x = *y;
*y = temp;
}
/***********************************************************************/
/*          Random value generator: Generates a value within bounds     */
/***********************************************************************/
int randval(int low, int high)
{
int val;
int nb = high - low + 1;
val = (rand()%nb + low);
return(val);
}
/***********************************************************************/
/* routine to find sum of values */
float findsum(float vals[],int num_els)
{
float summation=0.0;
        for (int i=0;i<num_els; i++)      {
                summation=summation+vals[i];
        }
        return summation;
}
/***********************************************************************/
void get_f(char name[])
{
        int c_count,flag;
        flag = 1;
        while(flag)
        {
        gets(name);
        c_count = strlen(name) ;
        if (c_count > 80) puts("File name too long\n");
        else flag = 0;
        }
}

/***********************************************************************/
/* Allocation of vectors and matrices starts here */
int *ivector(int nl, int nh)   /* allocates an int vector with range [nl....nh] */
{
        int *v ;

        v = (int *)malloc((unsigned) (nh-nl+1)*sizeof(int)) ;
        if(!v) nrerror(" allocation failure in ivector()") ;
        return v-nl ;
}

/***********************************************************************/
float *vector(int nl, int nh)   /* allocates a vector with range [nl....nh] */
{
        float *v ;

        v = (float *)malloc((unsigned) (nh-nl+1)*sizeof(float)) ;
        if(!v) nrerror(" allocation failure in vector()") ;
        return v-nl ;
}
/*********************************************************************/
char *cvector(int nl, int nh)  /* allocates a vector with range [nl....nh] */
{
        char *v ;

        v = (char *)malloc((unsigned) (nh-nl+1)*sizeof(char)) ;
        if(!v) nrerror(" allocation failure in cvector()") ;
        return v-nl ;
}
/*********************************************************************/
void free_vector(float *v, int nl, int nh)      /* Frees a float vector allocated by vector */
{
        free((char*) (v+nl)) ;
}
/*********************************************************************/
void free_ivector(int *v, int nl, int nh)       /* Frees a float vector allocated by ivector */
{
        free((char*) (v+nl)) ;
}
/*********************************************************************/
float **matrix(int nrl, int nrh, int ncl, int nch)
{
        int i;
        float **m;

        m = (float **) malloc((unsigned) (nrh-nrl+1)*sizeof(float*)) ;
        if(!m) nrerror("allocation failure 1 in matrix()") ;
        m -= nrl ;

        for(i = nrl; i <= nrh; i++) {
                m[i] = (float *) malloc((unsigned) (nch-ncl+1)*sizeof(float)) ;
                if(!m[i]) nrerror("allocation failure 2 in matrix()") ;
                m[i] -= ncl ;
        }
        return m ;
}
/*********************************************************************/
int **imatrix(int nrl, int nrh, int ncl, int nch)
{
        int i;
        int **m;

        m = (int **) malloc((unsigned) (nrh-nrl+1)*sizeof(int*)) ;
        if(!m) nrerror("allocation failure 1 in matrix()") ;
        m -= nrl ;

        for(i = nrl; i <= nrh; i++) {
                m[i] = (int *) malloc((unsigned) (nch-ncl+1)*sizeof(int)) ;
                if(!m[i]) nrerror("allocation failure 2 in matrix()") ;
                m[i] -= ncl ;
        }
        return m ;
}
/* ************************************************** */
/* Routine to print error message */
void nrerror(char *messg)
{
        puts(messg) ; getchar() ;
        exit(1) ;
}
/*********************************************************************/
```

C. Discrete Differential Dynamic Programming Code

This appendix contains the DDDP code used to solve the four reservoir problem.

```
/*********************************************************************/
/*          Discrete Differential Dynamic Programming Code                    */
/*          for the Four Reservoir System 4r_DDDP.CPP                         */
/*********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define MAXSTATES1 11
#define MAXSTATES2 11
#define MAXSTATES3 11
#define MAXSTATES4 16
#define MINSTATES 0
/*********************************************************************/
#define MAX_ITER 20
#define num_reservoir 4
#define num_stages 12
/*********************************************************************/
struct reschar                    /* reservoir characteristics */
{
 float inflow[num_stages];        /* an array for inflows  */
 float stormin;
 float stormax;
 float storinc;
 float relinc;
 float relmin;
 float relmax;
};

struct reschar res[num_reservoir];   /* number of reservoirs */

int **imatrix(int nrl, int nrh, int ncl, int nch) ;
float **matrix(int nrl, int nrh, int ncl, int nch) ;
int *ivector(int nl, int nh) ;
```

```
float *vector(int nl, int nh) ;
char *cvector(int nl, int nh) ;

/*********************************************************************/
main()
{
int ii,rr,jj,kk,jj2,jj3,jj4,kk2,kk3,kk4,ins,ins2,ins3,ins4;
int is,stage_index,dummy;
float new_state1,new_state2,new_state3,new_state4;
float **ben;
float ret_funct[num_stages+1][12][12][12][17];
float r1,r2,r3,r4,Increase,max_return,cum_return,*ObjValue;;
float *res_storage,*res_storage2,*res_storage3,*res_storage4;

short int previous_state[num_stages+1][12][12][12][17];
short int previous_state2[num_stages+1][12][12][12][17];
short int previous_state3[num_stages+1][12][12][12][17];
short int previous_state4[num_stages+1][12][12][12][17];

int *kkmax,*kkmax2,*kkmax3,*kkmax4;
int lc1,uc1,lc2,uc2,lc3,uc3,lc4,uc4,sigma;
int ln1,un1,ln2,un2,ln3,un3,ln4,un4,jk,jk2,jk3,jk4;
int it,lastkk,lastkk2,lastkk3,lastkk4;
char title[80];
char benfuncfile[20];
char rescharfile[20];
char initrajfile[20];
char nlpoutfile[20];
FILE *f1,*f2,*f3;
/*********************************************************************/

        /* Open the main file */
        if(!(f1 = fopen("4r_idp12.lst", "r"))) {
        printf(" Can't open 4r_idp12.lst on f2/n") ;
        exit(1) ;
  }
        fgets(title, 80, f1);
        fscanf(f1,"%d\n",&dummy);
        if(num_stages!=dummy) {
                printf("check num_stages\n");
                exit(1);
        }
        puts(title);
```

```
          fgets(nlpoutfile,20,f1);
          fgets(benfuncfile,20,f1);
          puts(benfuncfile);
          fgets(rescharfile,20,f1);
          fgets(initrajfile,20,f1);

          for (ii=0; ii<20; ii++) {
                    if (benfuncfile[ii] =='\n') benfuncfile[ii]='\0';
                    if (rescharfile[ii] =='\n') rescharfile[ii]='\0';
                    if (initrajfile[ii] =='\n') initrajfile[ii]='\0';
                    if (nlpoutfile[ii] =='\n') nlpoutfile[ii]='\0';
          }
          (void)fclose(f1) ;
          /* Open an output file */
          if(!(f2 = fopen(nlpoutfile, "w"))) {
          printf(" Can't open nlpoutfile on f2/n") ;
          exit(1) ;
  }
/*********************************************************************/
          ObjValue = vector(0,MAX_ITER);
          res_storage = vector(0,MAXSTATES1-1);
          res_storage2 = vector(0,MAXSTATES2-1);
          res_storage3 = vector(0,MAXSTATES3-1);
          res_storage4 = vector(0,MAXSTATES4-1);
/*********************************************************************/
          kkmax = ivector(0,num_stages);
          kkmax2 = ivector(0,num_stages);
          kkmax3 = ivector(0,num_stages);
          kkmax4 = ivector(0,num_stages);
/*****************************************************************/
          /* Read benefit function values */
          if(!(f3 = fopen(benfuncfile, "r"))) {
          printf(" Can't open ben_func.dat on f3/n") ;
          exit(1) ;
  }
  ben = matrix(0, num_stages-1, 0, 4) ;
  fgets(title, 80, f3) ;
          puts(title);
  for(stage_index = 0; stage_index < num_stages; stage_index++) {
          ii = stage_index ;
          for(jj = 0; jj < 5; jj++) {
                    fscanf(f3,"%f",&ben[ii][jj]) ;
          }
  }
          (void)fclose(f3) ;
/**************************************************************/
          /* Read reservoir characteristics */
          if(!(f3 = fopen(rescharfile, "r"))) {
          printf(" Can't open test.dat on f3/n") ;
          exit(1) ;
           }
          fgets(title, 80, f3) ;
          for(rr = 0; rr < num_reservoir; rr++) {
                    fscanf(f3,"%f  %f %f" ,&res[rr].relmin,&res[rr].relmax,&res[rr].relinc) ;
          }

          for(rr = 0; rr < num_reservoir; rr++) {
                    fscanf(f3,"%f  %f %f" ,&res[rr].stormin,&res[rr].stormax,&res[rr].storinc) ;
          }

          (void)fclose(f3) ;
/*******************************************************************/
          /* assigns inflows to reservoir1 and reservoir 2 */
          for(stage_index = 0; stage_index < num_stages; stage_index++) {
                    ii = stage_index ;
                    res[0].inflow[ii] = 2.0F;
                    res[1].inflow[ii] = 3.0F;
          }
          printf("data read\n");
/*******************************************************************/
          for(kk = 0; kk < MAXSTATES1; kk++) {
                    res_storage[kk] = res[0].stormin + kk*res[0].storinc;
          }
/*******************************************************************/
          for(kk = 0; kk < MAXSTATES2; kk++) {
                    res_storage2[kk] = res[1].stormin + kk*res[1].storinc;
          }
/*******************************************************************/
          for(kk = 0; kk < MAXSTATES3; kk++) {
                    res_storage3[kk] = res[2].stormin + kk*res[2].storinc;
          }
/*******************************************************************/
          for(kk = 0; kk < MAXSTATES4; kk++) {
                    res_storage4[kk] = res[3].stormin + kk*res[3].storinc;
          }
```

```
/***********************************************************************/
          /* read initial trial state trajectories */
          if(!(f3 = fopen(initrajfile, "r"))) {
          printf(" Can't open init_tra.dat on f3/n") ;
          exit(1) ;
          }
          fgets(title, 80, f3) ;
          puts(title);
          for(stage_index = 0; stage_index <=num_stages; stage_index++) {
                    ii = stage_index ;
                    fscanf(f3,"%d %d %d %d", &kkmax[ii],&kkmax2[ii],&kkmax3[ii],&kkmax4[ii]) ;
          }
          (void)fclose(f3) ;
/*****************************************************************/
          it = 0;
          ObjValue[0]=362.50F;
          sigma = 1;
          Increase = 10.0F;
/***********************************************************************/
          /* start the DDDP iterative loop */
          while(Increase > 0.1F) {
          printf("it = %2d ObjValue[%d]= %3.2f\n", it,it,ObjValue[it]);
/*****************************************************************/
          /* initialise previous state and return function to zero */
          for(ii = 0; ii <= num_stages; ii++) {
          for(jj =0; jj < MAXSTATES1; jj++) {
                    for(jj2=0; jj2 < MAXSTATES2; jj2++) {
                                        for(jj3=0; jj3 < MAXSTATES3; jj3++) {
                                                  for(jj4=0; jj4 < MAXSTATES4; jj4++) {
                                                  ret_funct[ii][jj][jj2][jj3][jj4] = 0.0F ;
                                                            previous_state[ii][jj][jj2][jj3][jj4] = 0 ;
                                                            previous_state2[ii][jj][jj2][jj3][jj4] = 0 ;
                                                            previous_state3[ii][jj][jj2][jj3][jj4] = 0 ;
                                                            previous_state4[ii][jj][jj2][jj3][jj4] = 0 ;
                                                  }
                                        }
                    }
          }
  }
/***********************************************************************/
          /* locate the starting states */
          /* initial return function to a non-zero value */
          ret_funct[0][5][5][5][5] = 0.001F ;
```

```
/**********************************************************************/
            /* start DP loop */
  printf(" Start of DP Loop\n") ;
  for(stage_index = 0; stage_index < num_stages; stage_index++) {
            ii = stage_index ;
                        printf("ii=%d\n",ii);
                        /* define upper and upper bounds of state trajectory */
                        lc1 = kkmax[ii] - sigma; if (lc1<MINSTATES) lc1 = MINSTATES;
                        uc1=kkmax[ii] + sigma+1;          if (uc1>MAXSTATES1) uc1 = MAXSTATES1;
                        lc2 = kkmax2[ii] - sigma; if (lc2<MINSTATES) lc2 = MINSTATES;
                        uc2=kkmax2[ii] + sigma+1;         if (uc2>MAXSTATES2) uc2 = MAXSTATES2;
                        lc3 = kkmax3[ii] - sigma; if (lc3<MINSTATES) lc3 = MINSTATES;
                        uc3=kkmax3[ii] + sigma+1;         if (uc3>MAXSTATES3) uc3 = MAXSTATES3;
                        lc4 = kkmax4[ii] - sigma; if (lc4<MINSTATES) lc4 = MINSTATES;
                        uc4=kkmax4[ii] + sigma+1;         if (uc4>MAXSTATES4) uc4 = MAXSTATES4;

                        ln1 = kkmax[ii+1] - sigma; if (ln1<MINSTATES) ln1 = MINSTATES;
                        un1=kkmax[ii+1] + sigma+1;     if (un1>MAXSTATES1) un1 = MAXSTATES1;
                        ln2 = kkmax2[ii+1] - sigma; if (ln2<MINSTATES) ln2 = MINSTATES;
                        un2=kkmax2[ii+1] + sigma+1;    if (un2>MAXSTATES2) un2 = MAXSTATES2;
                        ln3 = kkmax3[ii+1] - sigma; if (ln3<MINSTATES) ln3 = MINSTATES;
                        un3=kkmax3[ii+1] + sigma+1;    if (un3>MAXSTATES3) un3 = MAXSTATES3;
                        ln4 = kkmax4[ii+1] - sigma; if (ln4<MINSTATES) ln4 = MINSTATES;
                        un4=kkmax4[ii+1] + sigma+1;    if (un4>MAXSTATES4) un4 = MAXSTATES4;
                        for(jk = ln1; jk < un1; jk++) {
                        for(kk = lc1; kk < uc1; kk++) {
                                    for(jk2 = ln2; jk2 < un2; jk2++) {
                                                for(kk2 = lc2; kk2 < uc2; kk2++) {
                                                            for(jk3 = ln3; jk3 < un3; jk3++) {
                                                                        for(kk3 = lc3; kk3 < uc3; kk3++) {
                                                                                    for(jk4 = ln4; jk4 < un4; jk4++) {
                                                                                    for(kk4 = lc4; kk4 < uc4; kk4++) {
                                                                                                if(ret_funct[ii][kk][kk2][kk3][kk4] > 0.0001F) {
                                                                                                            r1 = res_storage[kk]-res_storage[jk] + res[0].inflow[ii];
                                                                                                            if(r1>=res[0].relmin && r1<=res[0].relmax) {
                                                                                                new_state1 = res_storage[jk];

                                                                                                /* find the corresponding state */
                                                                                                ins = 0 ;
                                                                                                for(is = 1; is < MAXSTATES1; is++) {
                                                                                                            if(new_state1 >= res_storage[is-1] && new_state1<res_storage[is]) ins =
                                                                                                                                                            is-1  ;
                                                                                                }
```

```
if(new_state1 >= res[0].stormax) {
            ins = MAXSTATES1 - 1 ;
}
            r2 = res_storage2[kk2]-res_storage2[jk2] + res[1].inflow[ii];

if(r2>=res[1].relmin && r2<=res[1].relmax) {
            new_state2 = res_storage2[jk2];
            ins2 = 0 ;
            for(is = 1; is < MAXSTATES2; is++) {
                                    if(new_state2 >= res_storage2[is-1] && new_state2 <
res_storage2[is]) ins2 = is-1 ;
            }

            if(new_state2 >= res[1].stormax) {
            ins2 = MAXSTATES2 - 1 ;
            }

            res[2].inflow[ii]= r2;
            r3 = res_storage3[kk3]-res_storage3[jk3] + res[2].inflow[ii];
            if(r3 >= res[2].relmin && r3<=res[2].relmax) {
            new_state3 = res_storage3[jk3];
            ins3 = 0 ;
            for(is = 1; is < MAXSTATES3; is++) {
                                    if(new_state3 >= res_storage3[is-1] && new_state3 <
            res_storage3[is]) ins3 = is-1  ;
            }

            if(new_state3 >= res[2].stormax && r3<=res[2].relmax) {
            ins3 = MAXSTATES3 - 1 ;
            }

            res[3].inflow[ii]= r1 + r3;
            r4 = res_storage4[kk4]-res_storage4[jk4] + res[3].inflow[ii];
            if(r4 >= res[3].relmin && r4 <= res[3].relmax) {
            new_state4 = res_storage4[jk4];
            ins4 = 0 ;
            for(is = 1; is < MAXSTATES4; is++) {
                                    if(new_state4 >= res_storage4[is-1] && new_state4 <
res_storage4[is]) ins4 = is-1  ;
            }

            if(new_state4 >= res[3].stormax) {
            ins4 = MAXSTATES4 - 1 ;
```

```
                                                                                                }
                                                                                                        /* compute cummulative return */
                                                                                                cum_return = ret_funct[ii][kk][kk2][kk3][kk4] +
                                                                                                                ben[ii][0]*r1+ben[ii][1]*r2
                                                                                                                +ben[ii][2]*r3 + (ben[ii][3] + ben[ii][4])*r4;
                                                                                                                if(cum_return >= ret_funct[ii+1][ins][ins2][ins3][ins4])
                                                                                                        {
                                                                                                                ret_funct[ii+1][ins][ins2][ins3][ins4] = cum_return ;
                                                                                                                previous_state[ii+1][ins][ins2][ins3][ins4] = kk ;
                                                                                                                previous_state2[ii+1][ins][ins2][ins3][ins4] = kk2 ;
                                                                                                                        previous_state3[ii+1][ins][ins2][ins3][ins4] =
                                                                                                                kk3 ;
                                                                                                                        previous_state4[ii+1][ins][ins2][ins3][ins4] =
                                                                                                                kk4 ;
                                                                                                        } /* if cum_returnloop */
                                                                                                } /*new_state4 loop*/
                                                                                                } /*new_state3 loop */
                                                                                        } /* new_state2 loop */
                                                                                        }  /* new_state1 loop */
                                                                                        } /* ret_funct loop */
                                                                                        } /* state loop for fourth */
                                                                                } /* decision loop for fourth res */
                                                                        }  /* state loop for third res */
                                                                } /* decision loop for third res */
                                                        }  /* state loop for second reservoir */
                                                }  /* decision loop for second reservoir */
                                        }  /* state loop for first reservoir */
                                } /* decision loop for first reservoir */
                        }  /* stage loop */
/*********************************************************************/
/**********************************************************************/
/*              trace back to get the optimal path                                      */
/**********************************************************************/
                printf("Trace back\n");
        for(stage_index = num_stages; stage_index > 0; stage_index--) {
                                max_return=0.0F;
                                ii = stage_index ;
                                if(ii == num_stages ) {
                                        /* constraints on ending storages */
                                        kkmax[ii] = 5 ; kkmax2[ii]= 5 ;
                                        kkmax3[ii]= 5 ;  kkmax4[ii]= 7 ;
                                }
        /* trace back to previous time step to determine feasible policies
```

```
            do by taking maximum return at each stage to cut down no looked at  */
            kk = kkmax[ii] ; kk2 = kkmax2[ii] ; kk3 = kkmax3[ii] ; kk4 = kkmax4[ii] ;
            kkmax[ii-1] = previous_state[ii][kk][kk2][kk3][kk4] ; kkmax2[ii-1] = previous_state2[ii][kk][kk2][kk3][kk4] ;
                        kkmax3[ii-1] = previous_state3[ii][kk][kk2][kk3][kk4] ;  kkmax4[ii-1]= previous_state4[ii][kk][kk2][kk3][kk4] ;
             }
             printf("TRACEBACK OK \n");

/***********************************************************************/
             /* Compute Objective function value for the current iteration*/
            lastkk = kkmax[num_stages]; lastkk2 = kkmax2[num_stages];
            lastkk3 = kkmax3[num_stages]; lastkk4 = kkmax4[num_stages];
            ObjValue[it+1] = ret_funct[num_stages][lastkk][lastkk2][lastkk3][lastkk4];
            Increase =(float)fabs(ObjValue[it+1] -ObjValue[it]) ;
            it++;
            } /* while loop */
            /* end of iterative loop of DDDP*/

            printf("Solution Converged\n");
            /* print optimum states and releases to the output file */

            fprintf(f2," PROGRAM FILE NAME: 4R_DDDP.CPP\n") ;
            fprintf(f2," OUTPUT FILE NAME : 4R_DDDP.OUT\n") ;
            fprintf(f2,"Stage  Str1  Inf1      Rel1  Str2  Inf2      Rel2  Str3  Inf3  Rel3  Str4  Inf4  Rel4  Return\n") ;

            for(ii = 0; ii < num_stages ; ii++) {
            kk = kkmax[ii+1] ;   kk2 = kkmax2[ii+1] ;
                        kk3 = kkmax3[ii+1] ;   kk4 = kkmax4[ii+1] ;
                        r1 =res_storage[kkmax[ii]] -res_storage[kkmax[ii+1]] +res[0].inflow[ii];
                        r2 =res_storage2[kkmax2[ii]] -res_storage2[kkmax2[ii+1]] +res[1].inflow[ii];
                        res[2].inflow[ii] = r2;
                        r3 =res_storage3[kkmax3[ii]] -res_storage3[kkmax3[ii+1]] +res[2].inflow[ii];
                        res[3].inflow[ii] = r1+r3;
                        r4 =res_storage4[kkmax4[ii]] -res_storage4[kkmax4[ii+1]] +res[3].inflow[ii];
                        fprintf(f2,"%3d  %5.2f  %5.2f %5.2f  %5.2f  %5.2f %5.2f  %5.2f  %5.2f  %5.2f  %5.2f  %5.2f  %5.2f  %7.2f\n",
                                    ii, res_storage[kkmax[ii]], res[0].inflow[ii], r1,
                                    res_storage2[kkmax2[ii]], res[1].inflow[ii], r2,
                                    res_storage3[kkmax3[ii]], res[2].inflow[ii], r3,
                                    res_storage4[kkmax4[ii]], res[3].inflow[ii], r4,
                                    ret_funct[ii+1][kk][kk2][kk3][kk4]) ;
            }
            (void)fclose(f2);
            printf("RESULTS PRINTED TO 4R_DDDP.OUT\n");
            return 0;
```

```
}           /* end of main function */
/**********************************************************************/
/*          Dynamic allocation of vectors and matrices starts here              */
int *ivector(int nl, int nh)   /* allocates an int vector with range [nl....nh] */
{
            int *v ;

            v = (int *)malloc((unsigned) (nh-nl+1)*sizeof(int)) ;
            if(!v) printf(" allocation failure in ivector()") ;
            return v-nl ;
}
/**********************************************************************/
float *vector(int nl, int nh)   /* allocates a vector with range [nl....nh] */
{
            float *v ;

            v = (float *)malloc((unsigned) (nh-nl+1)*sizeof(float)) ;
            if(!v) printf(" allocation failure in vector()") ;
            return v-nl ;
}

char *cvector(int nl, int nh)   /* allocates a vector with range [nl....nh] */
{
            char *v ;

            v = (char *)malloc((unsigned) (nh-nl+1)*sizeof(char)) ;
            if(!v) printf(" allocation failure in cvector()") ;
            return v-nl ;
}
/**********************************************************************/
void free_vector(float *v, int nl, int nh)        /* Frees a float vector allocated by vector */
{
            free((char*) (v+nl)) ;
}
/**********************************************************************/
void free_ivector(int *v, int nl, int nh)        /* Frees a float vector allocated by ivector */
{
            free((char*) (v+nl)) ;
}
/**********************************************************************/
float **matrix(int nrl, int nrh, int ncl, int nch)
{
            int i;
```

```
	float **m;

	m = (float **) malloc((unsigned) (nrh-nrl+1)*sizeof(float*)) ;
	if(!m) printf("allocation failure 1 in matrix()") ;
	m -= nrl ;

	for(i = nrl; i <= nrh; i++) {
		m[i] = (float *) malloc((unsigned) (nch-ncl+1)*sizeof(float)) ;
		if(!m[i]) printf("allocation failure 2 in matrix()") ;
		m[i] -= ncl ;
	}
	return m ;
}
/**********************************************************************/
int **imatrix(int nrl, int nrh, int ncl, int nch)
{
	int i;
	int **m;

	m = (int **) malloc((unsigned) (nrh-nrl+1)*sizeof(int*)) ;
	if(!m) printf("allocation failure 1 in matrix()") ;
	m -= nrl ;

	for(i = nrl; i <= nrh; i++) {
		m[i] = (int *) malloc((unsigned) (nch-ncl+1)*sizeof(int)) ;
		if(!m[i]) printf("allocation failure 2 in matrix()") ;
		m[i] -= ncl ;
	}
	return m ;
}
/**********************************************************************/
```

References

Adeli, H., and Cheng, N. T. (1993). "Integrated genetic algorithm for optimization of space structures." *J. Aerosp. Engrg.*, ASCE, 6(4), 315-328.

Adeli, H., and Cheng, N. T. (1994). "Concurrent genetic algorithm for optimization of large structures." *J. Aerosp. Engrg.*, ASCE, 7(6), 276-296.

Allen, R. B., and Bridgeman, S. G. (1986). "Dynamic programming in hydropower scheduling." *J. Water Resour. Plang. and Mgmt.*, ASCE, 112(3), 339-353.

Archibald, T. W., McKinnon, K. M., and Thomas, L. C. (1997). "An aggregate stochastic dynamic programming model of multireservoir systems." *Water Resour. Res.*, 33(2), 333-340.

Becker, L., and Yeh, W-G. W. (1974). "Optimization of real time operation of multiple-reservoir system." *Water Resour. Res.*, 10(6), 1107-1112.

Becker, L., Yeh, W-G. W., Fults, D., and Sparks, D. (1976). "Operations models for the Central Valley Project." *J. Water Resour. Plang. and Mgmt.*, ASCE, 102(WR1), 101-115.

Bellman, R. (1957). *Dynamic programming*, Princeton University Press, Princeton, New Jersey.

Bellman, R., and Dreyfus, S. (1962). *Applied dynamic programming*, Princeton University Press, Princeton, New Jersey.

Bhaskar, N. R., and Whitlach, E. E. (1980). "Derivation of monthly reservoir release policies." *Water Resour. Res.*, 16(6), 987-993.

Carny, V. (1985). "A thermodynamical approach to the travelling salesman problem." *J. Optimization Theory and Applications.*, 45, 41-51.

Cheu, R. L., Jin, X., Ng, K-C., Ng, Y-L., and Srinivasan, D. (1998). "Calibration of FREESIM for Singapore Expressway using genetic algorithm." *J. Transp. Engrg.*, ASCE, 124(6), 526-536.

Chow, V. T., and Cortes_Rivera, G. (1974). "Applications of DDDP in water resources planning." Rep. 78, Univ. of Ill., Water Resources Center, Urbana.

Chow, V. T., Maidment, D. R., and Tauxe, G. W. (1975)." Computer time and memory requirements for DP and DDDP in water resources systems analysis." *Water Resour. Res.*, 11(5), 621-628.

Cieniawski, S. E., Eheart, J. W., and Ranjithan, S. (1995). "Using genetic algorithms to solve a multiobjective groundwater monitoring problem." *Water Resour. Res.*, 31(2), 399-409.

Collins, M. A. (1977). "Implementation of an optimization model for operation of a metropolitan reservoir system." *Water Resour. Bull.*, 13(1), 57-70.

Crawley, P. D., and Dandy, G. C. (1993). "Optimal operations of multiple-reservoir systems." *J. Water Resour. Plang. and Mgmt.*, ASCE, 119(1), 1-17.

Cunha, M. C., and Sousa, J. (1999). "Water distribution network design optimization: simulated annealing approach." *J. Water Resour. Plang. and Mgmt.*, ASCE, 125(4), 215-221.

Dandy, G. C., Simpson, A. R., and Murphy, L. J. (1996). "An improved genetic algorithm for pipe network optimization." *Water Resour. Res.*, 32(2), 449-458.

Darwin, C. (1859). *Origin of species.*

Dasgupta, D., and Michalewicz, Z. (1997). *Evolutionary algorithms in engineering applications.* Springer-Verlag, Berlin, Germany.

Davidson, J. W., and Goulter, I. C. (1995). "Evolution program for the design of rectilinear branched distribution systems." *J. Comp. in Civ. Engrg.*, ASCE, 9(2), 112-121.

Davis, L. (1991). *Handbook of genetic algorithms.* Van Nostrand, Reinhold.

De Jong, K. A. (1975). "An analysis of the behaviour of a class of genetic adaptive systems." PhD thesis, Univ. of Michigan., Ann Arbor, Mich.

Diaz, G. E., and Fontane, D. G. (1989). "Hydropower optimization via sequential quadratic programming." *J. Water Resour. Plang. and Mgmt.*, ASCE, 115(6), 715-733.

Dorfman, R. (1962). Mathematicl models: The multistructure approach, in *Design of water resources systems,* A. Maass, ed., Harward University Press, Cambridge, Mass.

Dudley, N. J., and Burt, O. R. (1973). "Stochastic reservoir management and system design for irrigation." *Water Resour. Res.*, 9(3), 507-522.

Esat, V., and Hall, M. J. (1994). "Water resources system optimisation using genetic algorithms." *Hydroinformatics '94, Proc., 1st Int. Conf. on Hydroinformatics,* Balkema, Rotterdam, The Netherlands, 225-231.

Fahmy, H. S., King, J. P., Wentzel, M. W., and Seton, J. A. (1994). "Economic optimization of river management using genetic algorithms." *Paper No. 943034, ASAE 1994 Int. Summer Meeting*, Am. Soc. of Agricultural Engrs., St. Joseph, Mich.

Fiering, M. B. (1967). *Streamflow synthesis.* Harvard University Press, Cambridge, Mass.

Feng, C-W., Liu, L., and Burns, S. A. (1997). "Using genetic algorithms to solve construction time-cost trade-off problem." *J. Comp. in Civ. Engrg.*, ASCE, 11(3), 184-189.

Fleming, G. (1975). *Computer simulation techniques in hydrology.* Elsevier, New York.

Franchini, M. (1996). " Use of a genetic algorithm combined with a local search method for the automatic calibration of conceptual rainfall-runoff models.", *J. Hydro. Sci.*, 41(1), 21-40.

Fwa, T. F., Chan, W. T., and Tan, C. Y. (1996). "Genetic algorithm programming of road maintenance and rehabilitation." *J. Transp. Engrg.*, ASCE, 112(3), 246-253.

Georgakakos, A. P., and Marks, D. H. (1987). "A new method for the real time operation of reservoir systems." *Water Resour. Res.*, 23(7), 1376-1390.

Georgakakos, A. P. (1989). "Extended linear quadratic Gaussian (ELQG) control: further extensions." *Water Resour. Res.*, 25(2), 191-201.

Georgakakos, A. (1996). A decision support system for the Equatorial Lakes, Lecture notes, First regional workshop, Lake Victoria Water Resources FAO Trust Fund Project GCP/RAF/304/JPN.

Georgakakos, A. P., Huaming, Y., and Yongqing, Y. (1997). "Control model for hydroelectric energy value optimization." *J. Water Resour. Plang. and Mgmt.*, ASCE, 123(1), 30-38.

Gilbert, K. C., and Shane, R. M. (1982). "TVA hydro scheduling model: theoretical aspects." *J. Water Resour. Plang. and Mgmt.*, ASCE, 108(WR1), 21-36.

Goldberg, D. E., and Kuo, C. H. (1987). "Genetic algorithms in pipeline optimization." *J. Comp. in Civ. Engrg.*, ASCE, 1(2), 128-141.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Reading, Mass.

Goldberg, D. E., and Deb, K. (1991). "A Comparative analysis of selection schemes used in genetic algorithms." Foundations of Genetic Algorithms, Morgan Kaufman, San Mateo, Calif., 69-93.

Grygier, J. C., and Stedinger, J. R. (1985). "Algorithms for optimizing hydropower system operation." *Water Resour. Res.*, 21(1), 1-10.

Guibert, J. A. T., Stedinger, J. R., and Staschus, K. (1990). "Optimization of value of CVP's hydropower production." *J. Water Resour. Plang. and Mgmt.*, ASCE, 116(1), 52-70.

Halhal, D., Walters, G. A., Ouazar, D., and Savic, D. A. (1997). "Water network rehabilitation with structured genetic algorithm." *J. Water Resour. Plang. and Mgmt.*, ASCE, 123(3), 137-146.

Hall, W. A., and Buras, N. (1961). "The dynamic programming approach to water resources development." *J. Geophys. Res.*, 66(2), 510-520.

Hall, W. A., Harboe, W., Yeh, W. W-G., and Askew, A. J. (1969). "Optimum firm power output from a two reservoir system by incremental dynamic programming." *Water Resour. Center Contrib.*, University of Calif., 130, 273-282.

Hall, M. J., and Savenije, H. H. (1995). "Flexibility and formality in the planning and management of water resources systems." British Hydrological Society, Fifth National Hydrology Symposium, Edinburgh.

Heidari, M., Chow, V. T., Kokotovic, P. V., and Meredith, D. D. (1971). "Discrete differential dynamic programming approach to water resources systems optimization." *Water Resour. Res.*, 7(2), 273-283.

Hiew, K-L. (1987). "Optimization algorithms for large scale multi reservoir hydropower systems." PhD thesis, Colorado State Univ., Fort collins, Colo.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* MIT Press, Cambridge, Mass.

Hollstien, R. B. (1971). "Artificial genetic adaptation in computer control systems." *Diss. Abstr. Int. B*, 32(3), 1510.

Hooper, E. R., Georgakakos, A. P., and Lettenmaier, D. P. (1991). "Optimal stochastic operation of Salt River Project, Arizona." *J. Water Resour. Plang. and Mgmt.*, ASCE, 117(5), 567-587.

Howson, H. R., and Sancho, N. G. F. (1975). "A new algorithm for the solution of multistate dynamic programming problems." *Math. Programming*, 8(1), 104-116.

Hurst, H. E. (1951). "Long-term storage capacity of reservoirs." *Trans.*, ASCE, 116, 770-808.

Hydrological Engineering Ceneter, HEC 5 (1979). Reservoir system operation for flood control and conservation user manual, technical report, U.S., Army Corps of Eng., Davis, Calif.

Institute of Hydrology. (1993). A review and update of the water balance of lake Victoria in East Africa., ODA Report 93/3.

Jacobson, D., and Mayne, D. (1970). *Differential dynamic programming.* Elsevier, New York.

Jain, S. K., Goel, M. K., and Agarwal, P. K. (1998). "Reservoir operation studies of Sabarmati system, India." *J. Water Resour. Plang. and Mgmt.*, ASCE, 124(1), 31-38.

Japanese Overseas Technical Co-operation Agency. (1973). The Brantas river basin development plan. Govt. of Indonesia, Directorate General of Water Resources Development, Indonesia.

Japan International Co-operation Agency. (1985). Widas flood control and drainage project, Part I study. Govt. of Indonesia, Directorate General of Cipta Karya.

Jenkins, W. M. (1991). "Structural optimisation using genetic algorithms." *The Struct. Engr.*, London, England, 69(24), 418-422.

Jenkins, W. M. (1992). "Plane frame optimum design environment based on genetic algorithms." *J. Struct. Engrg.*, ASCE, 118(11), 3103-3112.

Karamouz, M., and Houcks, M. H. (1982). "Annual and monthly reservoir operating rules." *Water Resour. Res.*, 18(5), 1333-1344.

Karamouz, M., Houck, M. H., and Delleur, J. W. (1992). "Optimization and simulations of multiple reservoir systems." *J. Water Resour. Plang. and Mgmt.*, ASCE, 118(1), 71-81.

Kennedy and Donkin. (1996). Hydropower development master plan - Part 1. Uganda Electricity Board, Uganda.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). "Optimization by simulated annealing." *Sci.*, 220(4598), 671-680.

Klemes, V. (1974). "The Hurst phenomenon : A puzzle." *Water Resour. Res.*, 10(4), 675-688.

Koumousis, V. K., and Georgiou, P. G. (1994). "Genetic algorithms in discrete optimization of steel truss roofs." *J. Comp. in Civ. Engrg.*, ASCE, 8(3), 309-325.

Kuo, J-T., Hsu, N-S., Chu, W-S., Shian, W., and Lin, Y-J. (1990). "Real-time operation of Tanshui river reservoirs." *J. Water Resour. Plang. and Mgmt.*, ASCE, 116(3), 349-361.

Labadie, J. W., Fontane, D. G., Tabios, G. Q., and Chou, N. F. (1987). "Stochastic analysis of dependable hydropower capacity." *J. Water Resour. Plang. and Mgmt.*, ASCE, 113(3), 422-437.

Larson, R. E. (1968). *State increment dynamic programming*, American Elsevier, New York.

Lee, H-L., Liebman, J. C., and Brill Jr, E. D. (1992). "Performance evaluation of Lake Shelbyville by stochastic dynamic programming." *J. Water Resour. Plang. and Mgmt.*, ASCE, 118(2), 185-204.

Lee, M. A., and Takagi, H. (1993). "Dynamic control of genetic algorithms using fuzzy logic techniques." *Hydroinformatics '94, Proc., 5th Int. Conf. on Genetic Algorithms,* M. Kaufmann, San Mateo, Calif., 76-83.

Lee, E. S., Waziruddin, S. (1970). "Applying gradient projection and conjugate gradient to the optimum operation of reservoirs." *Water Resour. Bull.*, 6(5), 713-724.

Li, G., and Mays, L. W. (1995). "Differential dynamic programming for estuarine management." *J. Water Resour. Plang. and Mgmt.*, ASCE, 121(6), 455-462.

Li, H., and Love, P. E. D. (1998). "Site-level facilities layout using genetic algorithms." *J. Comp. in Civ. Engrg.*, ASCE, 12(4), 227-231.

Little, J. D. C. (1955). "The use of storage water in a hydroelectric system." *Oper. Res.*, 3, 187-197.

Liu, C. L., Hammond, A., and Itoh, Y. (1997). "Maintenance strategy optimization of bridge decks using genetic algorithm." *J. Transp. Engrg.*, ASCE, 123(2), 91-100.

Loucks, D. P., and Falkson, L. M. (1970). "A comparison of some dynamic, linear and policy iteration methods for reservoir operation." *Water Resour. Bull.*, 6(3), 384-400.

Loucks, D. P., Stedinger, J. R., and Haith, D. (1981). *Water resources systems planning and analysis*, Prentice Hall, Englewood Cliffs, N. J.

Mandelbrot, B. B., and Wallis, J. R. (1968). "Noah, Joseph, and operational hydrology." *Water Resour. Res.*, 4(5), 909-917.

Marino, M. A., and Mohammadi, B. (1983). "Reservoir operation by linear and dynamic programming." *J. Water Resour. Plang. and Mgmt.*, ASCE, 109(4), 303-319.

Marino, M. A., and Loaiciga, H. A. (1985a). "Dynamic model for multireservoir operation." *Water Resour. Res.*, 21(5), 619-630.

Marino, M. A., and Loaiciga, H. A. (1985b). "Quadratic model for reservoir management: application to the Central Valley Project." *Water Resour. Res.*, 21(5), 631-641.

Martin, Q. W. (1995). "Optimal reservoir control for hydropower on Colorado River, Texas." *J. Water Resour. Plang. and Mgmt.*, ASCE, 121(6), 438-446.

Mawer, P. A., and Thorn, D. (1974). "Improved dynamic programming procedures and their application to water resource systems." *Water Resour. Res.*, 10(2), 183-190.

Mays, L. W., and Tung, Y. K. (1992). *Hydrosystems engineering and management.* McGraw-Hill Book Co., Inc., New York.

McKinney, D. C., and Lin, M-D. (1994). "Genetic algorithm solution of groundwater management models." *Water Resour. Res.*, 30(6), 1897-1906.

McLaughlin, D., and Velasco, H. L. (1990). "Real-time control of a system of large hydropower reservoirs." *Water Resour. Res.*, 26(4), 623-635.

Michalewicz, Z. (1992). *Genetic Algorithms + data Structures = evolution programs,* Springer-Verlag, New York, Inc., New York.

Mizyed, N. R., Loftis, J. C., and Fontane, D. G. (1992). "Operation of large multireservoir systems using optimal control theory." *J. Water Resour. Plang. and Mgmt.*, ASCE, 118(4), 371-385.

Mott MacDonald. (1981). Surabaya Water Use Study, Govt. Of Indonesia, Directorate General of Cipta Karya.

Mott MacDonald. (1988). East Java Provincial Water Resources Master Plan Study for Water Supply, The Brantas Basin Simulation Model Final Report,. Annexe V, Indonesia.

Murray, D. M., and Yakowitz, S. (1979). "Constrained differential dynamic programming and its application to multiresevoir control." *Water Resour. Res.*, 15(5), 1017-1027.

Murphy, L. J., Simpson, A. R., and Dandy, G. C. (1993). "Design of a network using genetic algorithms." *Water,* 20, 40-42.

Navon, R., and McCrea, A. M. (1997). "Selection of optimal construction robot using genetic algorithms." *J. Comp. in Civ. Engrg.*, ASCE, 11(3), 175-183.

Nippon Koei Co. Ltd. (1961). Comprehensive report on the Kali Brantas overall project. Govt. of Indonesia, Ministry of Public Works and Electric Power.

Nippon Koei Co. Ltd. (1978). Karangkates project study report for the operation of Karangkates - Lahor reservoir, Govt. of Indonesia, Ministry of Public Works and Electric Power.

Ng, K-C. (1995). "Switching control systems and their design automation via genetic algorithms." PhD thesis, Univ. of Glasgow, Glasgow.

Nopmongcol, P., and Askew, A. J. (1976). "Multilevel incremental dynamic programming." *Water Resour. Res.*, 12(6), 1291-1297.

O' Connnell, P. E. (1971). "A simple stochastic modelling of Hurst's law." *Proc., Int. Symp. on Math. Models in Hydrology*, Int. Assoc. of Sci. Hydrol., Warsaw.

O' Connell, P. E. (1974). "Stochastic modelling of long-term persistence in streamflow sequences." PhD thesis, Imperial College, London.

Oliveira, R., and Loucks, D. P. (1997). "Operating rules for multireservoir systems." *Water Resour. Res.*, 33(4), 839-852.

Opricobic, S., and Djordjivic, B. (1976). "Optimal long-term control of a multi-purpose reservoir with indirect users." *Water Resour. Res.*, 12(6), 1286-1290.

Ozden, M. (1984). "A binary state DP algorithm for operation problems of multireservoir systems." *Water Resour. Res.*, 20(1), 9-14.

Palmer, R. N., and Holmes, K. J. (1988). "Operational guidance during droughts: Expert system approach." *J. Water Resour. Plang. and Mgmt.*, ASCE, 114(6), 647-666.

Papageorgiou, M. (1985). "Optimal multireservoir network control by the discrete maximum principle." *Water Resour. Res.*, 21(12), 1824-1830.

Pontryagin, L. S., Boltyanskii, V., Gamkrelidze, R., and Mishchenko, E. (1962). *The mathematical theory of optimal processes* . Wiley-Interscience, New York.

Poonambalam, K., and Adams, B. J. (1996). "Stochastic optimization of multireservoir systems using a heuristic algorithm: case study from India." *Water Resour. Res.*, 32(3), 733-741.

Rajeev, S., and Krishnamoorthy, C. S. (1992). "Discrete optimization of structures using genetic algorithms." *J. Struct. Engrg.*, ASCE, 118(5), 1233-1250.

Rajeev, S., and Krishnamoorthy, C. S. (1997). "Genetic algorithm-based methodologies for design optimization of trusses." *J. Struct. Engrg.*, ASCE, 123(3), 350-358.

Raman, H., and Chandramouli, V. (1996) "Deriving a general operating policy for reservoirs using neural network." *J. Water Resour. Plang. and Mgmt.*, ASCE, 122(5), 342-347.

Randall, D., Houck, M. H., and Wright, J. R. (1990). "Drought management of existing water supply system." *J. Water Resour. Plang. and Mgmt.*, ASCE, 116(1), 1-20.

Rechenberg, I. (1973). *Evolutionstrategie [Evolution Strategy],* Struttgart, Frommann - Holzboog.

Reznicek, K. K., and Simonovic, S. P. (1990). "An improved algorithm for hydropower optimization." *Water Resour. Res.*, 26(2), 189-198.

Ritzel, B. J., Eheart, J. W., and Ranjithan, S. (1994). "Using genetic algorithms to solve a multiple objective groundwater pollution problem." *Water Resour. Res.*, 30(5), 1589-1603.

Rosenthal, R. E. (1981). "A nonlinear flow algorithm for maximization of benefits in a hydroelectric power system." *Oper. Res.*, 29(4), 763-786.

Russell, S. O., and Campbell, P. F. (1996). "Reservoir operating rules with fuzzy programming." *J. Water Resour. Plang. and Mgmt.*, ASCE, 122(3), 165-170.

Saad, M., Turgeon, A., Bigras, P., and Duquette, R. (1994). "Learning disaggregation technique for the operation of long-term hydroelectric power systems." *Water Resour. Res.*, 30(11), 3195-3202.

Shahin, M. (1985). Hydrology of the Nile basin, in *Developments in water science 21,* Elsevier, New York.

Shane, R. M., and Gilbert, K. C. (1982). "TVA hydro scheduling model: Practical aspects." *J. Water Resour. Plang. and Mgmt.*, ASCE, 108(1), 21-36.

Shrestha, B. P., Duckstein, L., and Stakhiv, E. Z. (1996). "Fuzzy rule-based modeling of reservoir operation." *J. Water Resour. Plang. and Mgmt.*, ASCE, 122(4), 262-269.

Stedinger, J. R., Sule, B. F., and Loucks, D. P. (1984). "Stochastic dynamic programming models for reservoir operation optimization." *Water Resour. Res.*, 20(11), 1499-1505.

Soh, C. K., and Yang, J. P. (1996). "Fuzzy controlled genetic algorithm search for shape optimization." *J. Comp. in Civ. Engrg.*, ASCE, 10(2), 141-150.

Syswerda, G. (1989). "Uniform crossover in genetic algorithms." *Proc., 3rd Int. Conf. on Genetic Algorithms*, J. D. Schaffer, ed., Morgan Kaufman, San Francisco, 2-9.

Savic, D. A., and Walters, G. A. (1997). "Genetic algorithms for least-cost design of water distribution networks." *J. Water Resour. Plang. and Mgmt.*, ASCE, 123(2), 67-77.

Simonovic, S. P., and Marino, M. A. (1980). "Reliability programming in reservoir management, 1. Single multipurpose reservoir." *Water Resour. Res.*, 16(5), 844-848.

Simonovic, S. P. (1992). "Reservoir systems analysis: Closing gap betwen theory and practice." *J. Water Resour. Plang. and Mgmt.*, ASCE, 118(3), 262-280.

Simpson, A. R., Dandy, G. C., and Murphy, L. J. (1994). "Genetic algorithms compared to other techniques for pipe optimization." *J. Water Resour. Plang. and Mgmt.*, ASCE, 120(4), 423-443.

Takeuchi, K., and Moreau, D. H. (1974). "Optimal control of multiunit interbasin water resource system." *Water Resour. Res.*, 10(3), 407-414.

Thomas, H. A., and Fiering, M. B. (1962). Mathematicl synthesis of streamflow sequences for the analysis of river basins by simulation, in *Design of water resources systems,* A. Mass, ed., Harward University Press, Cambridge, Mass.

Torabi, M., and Mobasheri, F. (1973). "A stochastic dynamic programming model for the optimum operation of a multi-purpose reservoir." *Water Resour. Bull.*, 9(6), 1089-1099.

Trezos, T., and Yeh, W. W-G. (1987). "Use of stochastic dynamic programming for reservoir management." *Water Resour. Res.*, 23(6), 983-996.

Trott, W. J., and Yeh, W. W-G. (1973). "Optimization of multiple reservoir system." *J. Hydraul. Div.,* ASCE, 99(HY10), 1865-1884.

Turgeon, A. (1980). "Optimal operation of multireservoir power systems with stochastic inflows." *Water Resour. Res.*, 16(2), 275-283.

Turgeon, A. (1981a). "Optimal short-term hydropower scheduling from the principle of progressive optimality." *Water Resour. Res.*, 17(3), 481-486.

Turgeon, A. (1981b). "A decomposition method for the long-term scheduling of reservoirs in series." *Water Resour. Res.*, 17(6), 1565-1570.

Turgeon, A. (1982). "Incremental dynamic programming may yield nonoptimal solutions." *Water Resour. Res.*, 18(6),1599-1604.

Wang, Q. J. (1991). "The genetic algorithm and its application to calibrating conceptual rainfall-runoff models." *Water Resour. Res.*, 27(9), 2467-2471.

Wardlaw, R. B. (1993). "Water resources systems modelling for the Brantas Basin in East Java." *Proc., 4th National Hydrology Symposium,* Univ. of Wales, Wales, United Kingdom.

Wardlaw, R. B., Moore, D. N., and Barnes, J. M. (1997). An assessment of the potential of optimisation in real time irrigation management, in *Water: Economics, management and Demand,* M. Kay, T. Franks, and L. Smith, eds., Chapman and Hall, London.

Wardlaw, R. B. (1998). "Lake Victoria regulation studies." Rep., Water Resources Assessment Project (DANIDA funded)., Directorate of Water Development, Uganda.

Wasimi, S., and Kitanidis, P. K. (1983). "Real-time forecasting and daily operation of a multireservoir system during floods by linear quadratic Gaussian control." *Water Resour. Res.*, 19(6), 1511-1522.

Whitley, D. (1989). "The GENITOR algorithm and selection pressure: why rank best allocation of reproductive trials is best." *Proc. 3rd Int. Conf. on Genetic Algorithms*, J. D. Schaffer, ed., Morgan Kaufman, San Francisco.

Wurbs, R. A. (1993). "Reservoir-system simulation and optimization models." *J. Water Resour. Plang. and Mgmt.*, ASCE, 119(4), 455-472.

Yakowitz, S. (1982). "Dynamic programming applications in water resources." *Water Resour. Res.*, 18(4),673-696.

Yang, J. P., and Soh, C. K. (1997). "Structural optimization by genetic algorithms with tournament selection." *J. Comp. in Civ. Engrg.*, ASCE, 11(3), 195-200.

Yeh, W. W-G., Becker, L., and Chu, W. S. (1979). "Real time hourly reservoir operation." *J. Water Resour. Plang. and Mgmt.*, ASCE, 105(WR2), 187-203.

Yeh, W. W-G., and Becker, L. (1982). "Multiobjective analysis of multireservoir operations." *Water Resour. Res.*, 18(5),1326-1336.

Yeh, W. W-G. (1985). "Reservoir management and operation models: A state-of-the-art review." *Water Resour. Res.*, 21(12),1797-1818.

Young, G. K. (1967). "Finding reservoir operating rules." *J. Hydr. Div.*, ASCE, 93(HY6), 297-321.

Zhao, B., and Mays, L. W. (1995). "Estuary management by stochastic linear quadratic optimal control." *J. Water Resour. Plang. and Mgmt.*, ASCE, 121(5), 382-398.

Publications

The following papers have either been published or submitted for publication in journals or conferences.

Published in Journals

- Wardlaw, R., and Sharif, M. (1999) "Evaluation of genetic algorithms for optimal reservoir system operation." *J. Water Resour. Plang. and Mgmt.*, ASCE, 125(1), 25-33.

- Sharif, M., and Wardlaw, R. (1999). "Multi reservoir systems optimization using genetic algorithms - A case study." Submitted August 1999, *J. Comp. in Civ. Engrg.*, ASCE.

Presented in Conference

Sharif, M., and Wardlaw, R. (1998). "Optimisation of multi reservoir systems using genetic algorithms." *Postgraduate Seminar*, Heriott-Watt University, January 1998.

Evaluation of Genetic Algorithms for Optimal Reservoir System Operation

By Robin Wardlaw[1] and Mohd Sharif[2]

Abstract: Several alternative formulations of a genetic algorithm for reservoir systems are evaluated using the four-reservoir, deterministic, finite-horizon problem. This has been done with a view to presenting fundamental guidelines for implementation of the approach to practical problems. Alternative representation, selection, crossover, and mutation schemes are considered. It is 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. The real-value coding operates significantly faster than binary coding and produces better results. The known global optimum for the four-reservoir problem can be achieved with real-value coding. A nonlinear four-reservoir problem is considered also, along with one with extended time horizons. The results demonstrate that a genetic algorithm could be satisfactorily used in real time operations with stochastically generated inflows. A more complex ten-reservoir problem is also considered, and results produced by a genetic algorithm are compared with previously published results. The genetic algorithm approach is robust and is easily applied to complex systems. It has potential as an alternative to stochastic dynamic programming approaches.

INTRODUCTION

The study of genetic algorithms (GAs) originated in the mid 1970s (Holland 1975) and has developed into a powerful optimization approach. Excellent introductions to GAs are given by Goldberg (1989) and by Michalewicz (1992), and several recent papers give summaries of the essentials (e.g., Oliveira and Loucks 1997; and Savic and Walters 1997).

The literature describing the application of GAs to water resources problems is not abundant. 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 optimized by minimizing the sum of squares of differences between computed and observed discharges. Of ten optimization runs, eight were able to locate the global minimum. The values obtained from the other two runs were only marginally inferior to the global optimum. 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.

There have been several applications of GAs to pipe network problems. Goldberg (1987) used a GA for pipeline optimization. Murphy et al. (1993) developed a methodology for optimizing a water supply network using a simple GA. The objective was to find the combination of pipe sizes that minimized the cost of a water distribution network. Simpson et al. (1994) compared the performance of complete enumeration, nonlinear programming (NLP) and a GA for an example pipe network. They concluded that the GA was capable of finding acceptable solutions, although for the example considered it was not as fast as NLP. Davidson and Goulter (1995) used GAs to optimize the layout of a branched rectilinear network, such as a natural gas or water distribution system. An improved GA has been developed by Dandy et al. (1996) for pipe network cost optimization and was found to perform better than the traditional optimization methods and a simple GA. The solution found by improved GA for the New York City water supply network was the lowest cost design yet presented in the literature for that particular problem. Savic and Walters (1997) describe the development of the computer model GANET for the least-cost design of water distribution networks, again demonstrating that in certain cases GAs may yield better results than other optimization techniques.

Halhal et al. (1997) described a multiobjective optimization approach using capital cost and benefit as dual objectives to the problem of network rehabilitation. They used a structured messy GA (SMGA), which has some additional features like variable string length that increases during the evolution of designs. They compared the performance of SMGA with a standard GA, concluding that the SMGA was much better for a large network.

Ritzel et al. (1994) solved a multiobjective ground-water pollution problem using a GA. Cieniawski et al. (1995) dealt with the multiobjective optimal location of a network of ground-water monitoring wells under conditions of uncertainty. McKinney and Lin (1994) also solved a ground-water management model with GAs.

GAs have so far had very little application in reservoir systems optimization. Esat and Hall (1994) applied a GA to the four-reservoir problem. The objective was to maximize the benefits from power generation and irrigation water supply subject to constraints on storages and releases from the reservoirs. The paper by Esat and Hall showed the significant potential of GAs in water resources systems optimization, and clearly demonstrated the advantages of GAs over standard dynamic programming (DP) techniques in terms of computational requirements. Fahmy et al. (1994) also applied a GA to a reservoir system, and compared performance of the GA approach with that of dynamic programming. They concluded that GAs had potential in application to large river basin systems.

Oliveira and Loucks (1997) used a GA to evaluate operating rules for multireservoir systems, demonstrating that GAs can be used to identify effective operating policies. Significant benefits were perceived to lie in the freedom afforded by GAs in the definition of operating policies and their evaluation.

The primary objective of this paper is to explore the potential of alternative GA formulations in application to reservoir systems, and to deterministic finite-horizon problems in particular. This paper significantly extends the work of Esat and Hall (1994) and leads to consideration of the potential of GAs in real-time reservoir operation with stochastic inflow forecasts. The problem addressed here differs from that considered by Oliveira and Loucks (1997), who were concerned with the

[1]Sr. Lect., Dept. of Civ. and Envir. Engrg., Univ. of Edinburgh, Edinburgh EH9 3JN, U.K. E-mail: Robin.Wardlaw@ed.ac.uk

[2]Lect., Dept. of Civ. Engrg., Facu. of Engrg. and Technol., Jamia Millia Islamia, New Delhi, India.

Note: Discussion open until July 1, 1999. To extend the closing date one month, a written request must be filed with the ASCE Manager of Journals. The manuscript for this paper was submitted for review and possible publication on December 4, 1997. This paper is part of the *Journal of Water Resources Planning and Management*, Vol. 125, No. 1, January/February, 1999. ©ASCE, ISSN 0733-9496/99/0001-0025-0033/ $8.00 + $.50 per page. Paper No. 17268.

optimization of parameters in operating policies rather than with deterministic real-time reservoir releases.

GAs may be set up in many ways, but as yet there is little guidance in the literature on the type of formulation most appropriate for reservoir systems. This paper is intended to address that gap through consideration of the application of GAs to the well-known four-reservoir problem (Larson 1968), which has provided a benchmark for water resources system optimization algorithms. The four-reservoir problem has previously been solved using a GA by Esat and Hall (1994). Heidari et al. (1971) solved the four-reservoir problem by discrete differential dynamic programming. Little detail was given by Esat and Hall on their GA formulation, although it appears that they did not consider every time step of the problem in the same way as considered here. In this paper several different approaches to GA formulation are considered, along with a range of sensitivity analyses. The object has been to present GAs as a practical tool in reservoir system evaluation, and to examine the potential of different GA formulations for multireservoir problems. Consideration has also been given to more complex and nonlinear problems and to problems with long time horizons. GAs deal easily with nonlinear problems and are shown to be very robust.

BASIS OF GENETIC ALGORITHMS

A GA is a search algorithm based upon the mechanics of natural selection, derived from the theory of natural evolution. GAs simulate mechanisms of population genetics and natural rules of survival in pursuit of the ideas of adaptation. Indeed this has led to a vocabulary borrowed from natural genetics.

Goldberg (1989) identifies the following as the significant differences between GAs and more traditional optimization methods:

- GAs work with a coding of the parameter set, not with the parameters themselves.
- GAs search from a population of points, not a single point.
- GAs use objective function information, not derivatives or other auxiliary knowledge.
- GAs use probabilistic transition rules, not deterministic rules.

A GA is a robust method for searching the optimum solution to a complex problem, although it may not necessarily lead to the best possible solution. A GA generally represents a solution using strings (also referred to as chromosomes) of variables that represent the problem. In early GAs (Goldberg and Kuo 1987; Wang 1991) these strings were comprised of binary bits. In binary representation, the bits may encode integers, real numbers, sets, or whatever else is appropriate to the problem. Real-value coding is now proving more effective in many problems than binary coding (e.g., Oliveira and Loucks 1997).

Coding components of possible solutions into a chromosome is the first part of a GA formulation. Each chromosome is a potential solution and is comprised of a series of substrings or genes, representing components or variables that either form or can be used to evaluate the objective function of the problem. In a simple single-reservoir problem in which the objective function is related to reservoir releases x_i over months $i = 1, 12$, the chromosome would comprise 12 genes, each representing one month of reservoir storage. Each gene could be represented by a binary string mapped to the range of permissible values of x, or by the real value of x. The fitness of a chromosome as a candidate solution to a problem is an expression of the value of the objective function represented by it. It is also a function of the problem constraints and may be modified through the introduction of penalties when constraints are not satisfied.

A GA starts with a population of chromosomes, which are combined through genetic operators to produce successively fitter chromosomes. The genetic operators used in the reproductive process are selection, crossover, and mutation. Chromosomes in the population with high fitness values have a high probability of being selected for combination with other chromosomes of high fitness. Combination is achieved through the crossover of pieces of genetic material between selected chromosomes. Mutation allows for the random mutations of bits of information in individual genes. Through successive generations, fitness should progressively improve. Various schemes for selection, crossover, and mutation exist and are discussed below.

In application to water resources problems, chromosomes may be generated that fail to meet system constraints, such as continuity and component capacities. Each generated chromosome must therefore be checked against the system constraints. Chromosomes failing to meet the constraints could be excluded from subsequent participation in the evolutionary process, but this may lead to useful genetic material being lost. Alternatives to exclusion are successive regeneration of chromosomes until they meet constraints, or application of a penalty function to reduce the fitness of chromosomes failing to meet constraints. Many argue that the former approach, adopted by Esat and Hall (1994), disrupts the GA process and in effect requires many additional generations (Michalewicz 1992). In this paper a penalty function approach has been adopted. The penalty function is quadratic, based on the degree of constraint violation. This is particularly useful in dealing with long strings representing a time series in which constraints are violated in only a small part. Also, potentially good genetic material is permitted to contribute to subsequent generations when violations are small.

Representation Schemes

Traditionally GAs have used binary coding, in which a chromosome is represented by a string of binary bits that can encode integers, real numbers, or anything else appropriate to a problem. Binary strings are easy to operate on, and within any gene, binary representations can be mapped to values in the range feasible for the variable represented (Goldberg 1989). Following the operations of reproduction, the fitness of a particular chromosome is evaluated after the binary values are decoded back into their original form. Standard binary coding of variables permits large jumps in variable values between generations, which can lead to difficulty in converging to a good solution. This can be overcome to some extent through the use of Gray coding (Goldberg 1989) in which the binary representation of two adjacent variable values changes by only one binary digit. Discretization of the decision variable space is required with binary and with Gray coding. Gene lengths can be modified to permit various levels of precision, and satisfactory mapping of the variable space can normally be achieved.

An alternative approach to formulation of the GA is to use a representation appropriate to the components of the problem. Real-value chromosomes have been used with success by various authors (e.g., Oliveira and Loucks 1997). In a real-value representation, individual genes of a chromosome are initially allocated values randomly within the feasible limits of the variable represented. With a sufficiently large population of chromosomes, adequate representation will be achieved. There is a significant advantage in not wasting computer time on decoding for objective function evaluation, although a more careful approach to mutation is required. In real-value coding there is no discretization of the decision variable space. This is another advantage of this approach.

Selection Approaches

Selection is the procedure by which chromosomes are chosen for participation in the reproduction process. A popular approach has been fitness proportionate selection (Goldberg 1989), in which the probability p_i of an individual i being selected is given by

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \tag{1}$$

where f_i = fitness of individual i, and n = population size.

Whitley (1989) identifies population diversity and selective pressure as being the two most important issues in the genetic search process. An interesting discussion on this topic is provided by Yang et al. (1997). Population diversity is maintained by exploiting good individuals, while still exploring the rest of the search space. It is influenced by the degree to which the best individuals are favored, which may be termed selective pressure. A high selective pressure may lead to a rapid convergence—but convergence to a suboptimal solution—while a low selective pressure may result in a greater number of generations being required before convergence to an acceptable solution is achieved. A number of approaches exist that help to balance the influence of selective pressure and population diversity.

Various rank selection schemes are in use (Michalewicz 1992) that tend to ensure that good chromosomes have higher chances of being selected for the next generation. Ranking schemes operate by sorting the population on the basis of fitness values and then assigning a probability of selection based upon the rank. A constant selection differential is thus maintained between the best and the worst individuals in the population (Whitley 1989). A drawback is that information on the relative fitness of the individuals is not used. Goldberg and Deb (1990) have compared various selection schemes, and indicated a preference for the tournament selection scheme, also discussed by Yang et al. (1997). In tournament selection a group of individuals are chosen at random from the population, and the individual with the highest fitness is selected for inclusion in the next generation. The procedure is repeated until the appropriate number of individuals are selected for the new generation. The approach had originally been developed with groups of two individuals and was called binary tournament selection, but larger groups lead to greater diversity and a smoother progression to a solution. Tournament selection was used by Cieniawski et al. (1995) in their ground-water monitoring problem.

Crossover Approaches

The general theory behind the crossover operation is that, by exchanging important building blocks between two strings that perform well, the GA attempts to create new strings that preserve the best material from two parent strings. The number of strings in which material is exchanged is controlled by the crossover probability forming part of the parametric data. Goldberg (1989) and Michalewicz (1992) describe the following methods of crossover: (1) one-point crossover; (2) two-point crossover; and (3) uniform crossover.

Crossover occurs between two selected chromosomes with some specified probability, usually in the range of 0.5–1.00 (i.e., selected chromosomes have this probability of being used in crossover). In one-point crossover, a crossover point is selected at random at some point c in the chromosome length L (Fig. 1). Two new individuals are created by swapping all genes between positions c and L. In two-point crossover, genetic material between two positions chosen at random along the length of the chromosomes is exchanged. Uniform crossover operates on individual genes of the selected chromosomes, rather than on blocks of genetic material, and each gene is considered in turn for crossover or exchange.

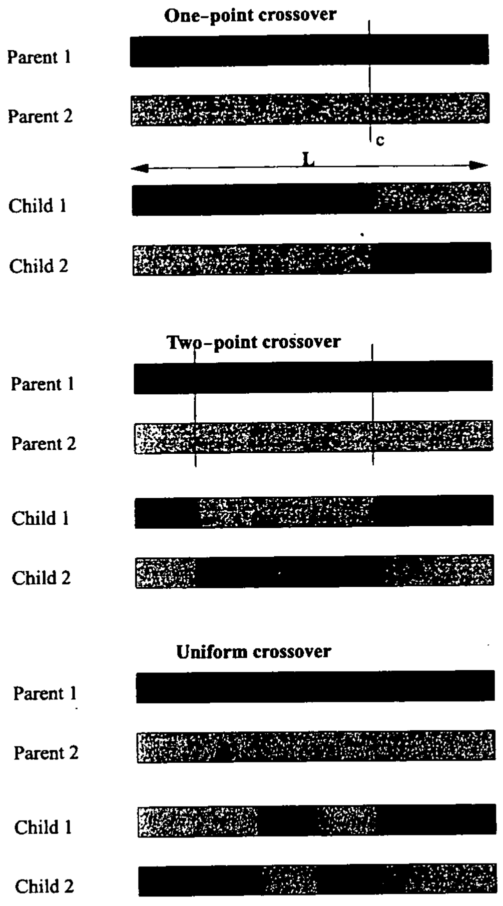


FIG. 1. Approaches to Crossover

An important aspect of crossover in application to a multivariate problem in binary coding is that crossover should occur only at gene boundaries, because each gene consists of alleles, or bits, and crossover may split the genes. This is not an issue for real-value representations. In real-value coding the gene comprises a single allele and is itself the parameter value.

Mutation Approaches

Mutation is an important process that permits new genetic material to be introduced to a population. A mutation probability is specified that permits random mutations to be made to individual genes. The two basic approaches to mutation for real-value representations are uniform mutation and nonuniform mutation (Michalewicz 1992). Uniform mutation permits the value of a gene to be mutated randomly within its feasible range of values, possibly resulting in significant modification of otherwise good solutions. In this study, a modified uniform mutation operator has been used. Modified uniform mutation permits modification of a gene by a specified amount, which may be either positive or negative. In nonuniform mutation, the amount by which genes are mutated can be reduced as a run progresses, and can therefore help in the later generations to fine tune the solutions. This operator is particularly suited to problems where high precision is required.

FOUR-RESERVOIR PROBLEM

The four-reservoir problem was formulated and first solved by Larson (1968), and more recently by Esat and Hall (1994). This problem offered the opportunity to test the performance of GAs against a known global optimum and to perform sensitivity analysis.

The system consists of four reservoirs, as shown in Fig. 2. The supplies from the system are used for hydropower generation and for irrigation. Hydropower generation is possible from each reservoir, and all discharges pass through the turbines. The outflow from reservoir four may be diverted for irrigation. Hydropower and irrigation benefits are quantified by linear functions of discharge. The objective is to maximize benefits from the system over 12 two-hour operating periods. There are inflows to the first and second reservoirs only, and these are 2 and 3 units, respectively, in all time periods. The initial storage in all reservoirs is 5 units.

The fundamental constraints on reservoir storage are

$$0.0 \le S_1, S_2, S_3 \le 10 \quad (2)$$

$$0.0 \le S_4 \le 15 \quad (3)$$

and on releases from the reservoirs through the turbines are

$$0.0 \le R_1 \le 3 \quad (4)$$

$$0.0 \le R_2, R_3 \le 4 \quad (5)$$

$$0.0 \le R_4 \le 7 \quad (6)$$

The above constraints apply in all time steps. The continuity constraints for each reservoir over each operating period t are

$$\mathbf{S}_i(t+1) = \mathbf{S}_i(t) + \mathbf{I}_i(t) + \mathbf{M}\cdot R_i(t) \quad (7)$$

where $\mathbf{S}_i(t)$ = vector of reservoir storages at time t in reservoirs $i = 1, n$; $\mathbf{I}_i(t)$ = vector of reservoir inflows in time period t to reservoirs $i = 1, n$; $\mathbf{R}_i(t)$ = vector of reservoir releases in time period t from reservoirs $i = 1, n$; and $\mathbf{M}$ = $n*n$ matrix of indices of reservoir connections

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & -1 \end{bmatrix}$$

In addition to the above general constraints, there are target final storages for all reservoirs. There are five units for reservoirs one to three, and seven units for reservoir four. The objective function to be maximized can be written as

$$\text{Max } I = \sum_{i=1}^{4}\sum_{t=0}^{11} b_i(t)\cdot R_i(t) + \sum_{t=0}^{11} b_5(t)\cdot R_4(t) \quad (8)$$

The benefit functions $b_i(t)$ are tabulated by Larson (1968) and by Heidari et al. (1971).

For comparative purposes, an LP solution to the above problem has been produced in a spreadsheet. This solution was identical to the original solution presented by Larson and has been used for evaluation of the GA approach discussed below. In the LP solution, the final target storages are set as constraints. In GA this is not possible, and use has been made of a penalty function (Heidari et al. 1971). Taking the target ending storage in reservoir i to be d_i, the penalty function is expressed as

$$g_i[S_i(12), d_i] = -40[S_i(12) - d_i]^2 \quad \text{for } S_i(12) \le d_i \quad (9)$$

and

$$g_i[S_i(12), d_i] = 0 \quad \text{for } S_i(12) > d_i \quad (10)$$

The objective function is thus modified to

$$\text{Max } I = \sum_{i=1}^{4}\sum_{t=0}^{11} b_i(t)\cdot R_i(t) + \sum_{t=0}^{11} b_5(t)\cdot R_4(t) + \sum_{i=1}^{4} g_i[S_i(12), d_i] \quad (11)$$

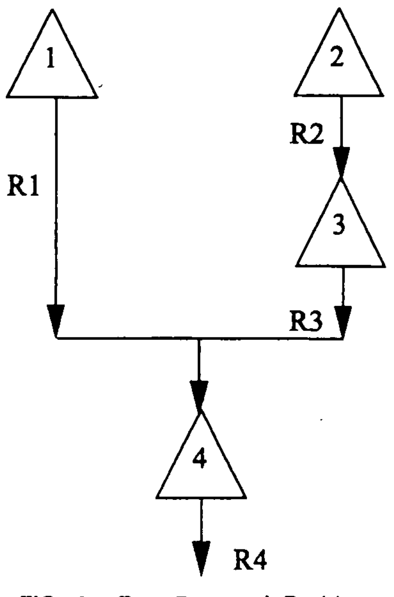


FIG. 2. Four-Reservoir Problem

FORMULATION OF GAs

To solve the four-reservoir problem using a GA, it is necessary to construct a chromosome representing all four reservoirs in all twelve time steps. Since the objective function is based on reservoir releases in each time step, releases should be the decision variable on which the GA is based. With four reservoirs and 12 time steps, there are thus 48 discrete variables to be represented in the GA. Each of these may be considered to be a gene, and in binary representation is encoded as a binary number. Reservoir releases are to be considered as integer quantities. This is toward defining the problem and is not a limitation of GAs. GAs work equally well with noninteger decision variables. In binary coding, three digits are required for the range of releases defined, and with this approach the total length of each chromosome in the population is thus 144. Esat and Hall (1994) refer to strings of length 16—it is unclear how these were made up. In real-value coding the length of a chromosome is 48 (12 time steps and 4 reservoirs).

The manner in which genes are grouped within the chromosome is of importance. There are two basic approaches. One would be to group releases by time step, such that the chromosome contained 12 groups of 4 genes representing the release from each reservoir in a particular time step. The alternative is to have 4 groups of 12 genes, with each group containing the time series of releases from an individual reservoir. The former approach is preferable, and keeps more closely related material together.

Consideration has been given to binary coding, Gray coding, and real-value coding. Performance of each of these codings and of different operator approaches is discussed next.

EVALUATION OF ALTERNATIVE GA FORMULATIONS

A series of sensitivity analyses were carried out to establish appropriate parameter settings under binary, Gray, and real-value codings, prior to evaluation of the alternative codings themselves. In many practical problems, GA results are found to be sensitive to crossover and mutation probabilities. This is because genetic material lost at the start of a run, through either crossover or mutation, may be needed to improve fitness in the later stages of a run. Sensitivity to crossover and mutation probability is discussed below for each coding scheme.

Sensitivity to Crossover Probability

Various researchers (e.g., DeJong 1975; Goldberg 1989) have suggested that good performance may be achieved from a GA using a high crossover probability and low mutation probability. It has been found from initial test runs that good results will be achieved when mutation is restricted to about one gene per chromosome on average. For the four-reservoir problem, with 48 genes, an initial mutation probability was therefore set at 0.02 (1/48).

Sensitivity to crossover probability was carried out using a population size of 100 and a mutation probability of 0.02. The tournament selection approach was adopted with uniform crossover, and a modified uniform mutation operator for real-valued representation. Crossover probabilities from 0.50 to 0.95 were considered through runs with a fixed length of 500 generations. Fig. 3 shows the sensitivity of the achieved fitness to crossover probability for each of the three coding schemes considered. Fitness is expressed as a proportion of the known optimum for the four-reservoir problem.

The real-value coding clearly provides the best performance over a wide range of crossover probabilities. The most appropriate crossover probability for the real-value code would appear to be in the range of 0.70–0.75. The optimal crossover probability for Gray coding appears to be 0.80. The binary coding is apparently more sensitive than either the real-value or Gray codings to crossover probability. There is a distinct peak in performance with a crossover probability of 0.70 as well as progressive deterioration in performance as crossover probability increases beyond this. The results demonstrate clearly that the GAs are robust, with reasonable results being obtained over a fairly wide range of crossover probabilities

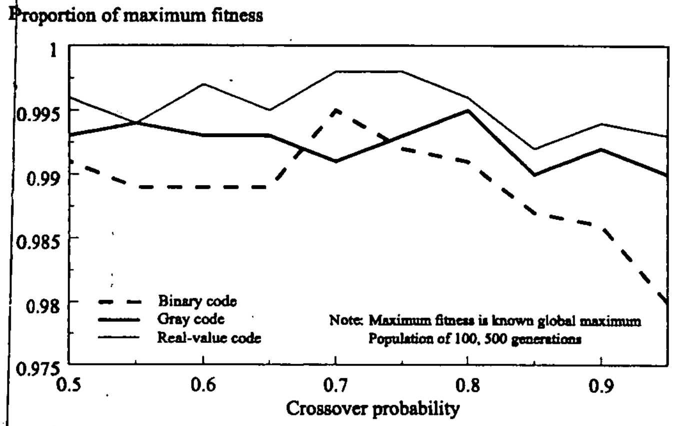


FIG. 3. Sensitivity to Crossover Probability

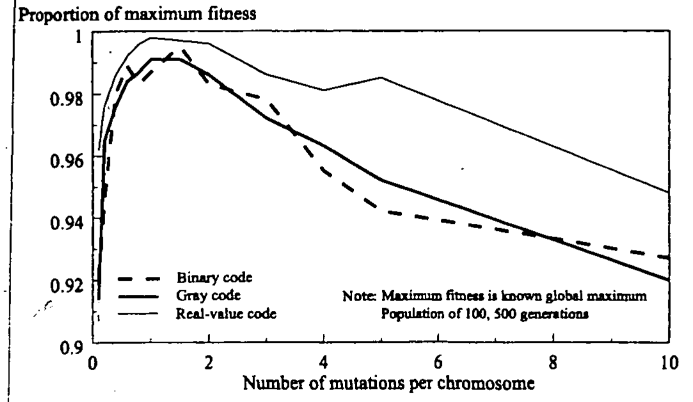


FIG. 4. Sensitivity to Mutation Probability

and coding methods. The results also demonstrate that the real-value representation is more robust and produces better results than either of the binary or Gray code representations.

Sensitivity to Mutation Probability

For the determination of sensitivity to mutation probability, a crossover probability of 0.70 was used, with all other parameters and run controls as discussed above. The mutation probability may be expressed as

$$P_{mut} = \frac{k}{C_{length}} \tag{12}$$

where P_{mut} = mutation probability; k = number of mutations; and C_{length} = chromosome length (in genes).

In the four-reservoir problem formulation used here, the length of a chromosome is 48 genes. Numbers of mutations per chromosome in the range of 0.1 to 10 have been considered, resulting in mutation probabilities of 0.002–0.208. The results of the sensitivity analysis are presented in Fig. 4. The best results are achieved with approximately one mutation per chromosome. In binary coding, the best result was achieved with 1.5 mutations per chromosome, but it is clear that there is more variability in the binary code representation. There is again an indication of robustness, and 0.6 to 2 mutations per chromosome produce reasonable results.

Representation Schemes

An evaluation of the three representation schemes has been carried out using the parameter levels that gave the best performance with each scheme in the sensitivity analysis. The results already presented in Figs. 3 and 4 have indicated that the real-value coding is likely to be the best. Fig. 5 presents a comparison of how maximum fitness values vary with generation number. Similar results are achieved with each scheme, although it is apparent that the real-value representation produces a smoother curve, perhaps indicating a greater robustness. The binary representation makes more erratic progress toward better fitness than do either the Gray code or real-value representations. The real-value representation exhibits a slower rate of improvement in fitness in the mid-generation period of the run, but it sustains its rate of improvement longer. The binary and Gray code representations become restricted partly because mutation may occur in any part of a gene, and while this is helpful in early generations it can be disruptive in later ones. More generations would be required to reach the optimum in binary or Gray coding, as the chromosome length is three times that of the real-value representation. Mutation can occur in any bit of the chromosome, and achieving an optimal sequence of 48 bits is quicker than achieving an optimal sequence of 144 bits.

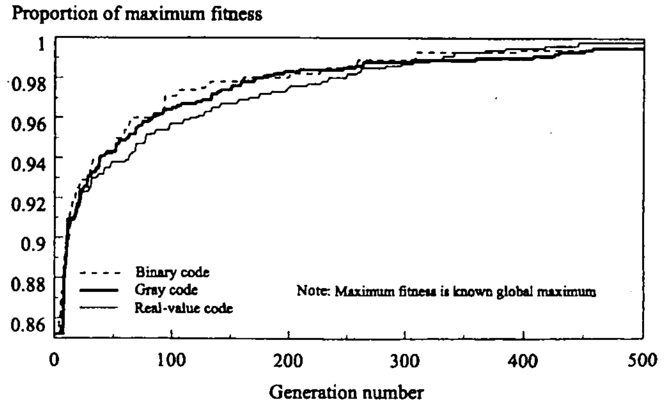


FIG. 5. Maximum Fitness with Alternative Representation Schemes

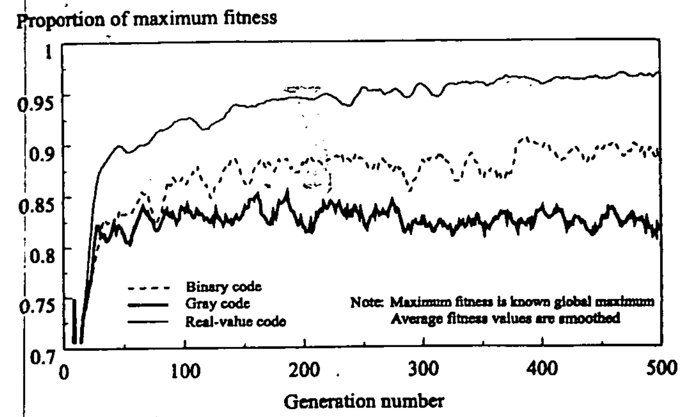


FIG. 6. Average Fitness with Alternative Representation Schemes

TABLE 1. Normalized and Actual Fitness Values after 500 Generations

Coding (1)	Normalized fitness (2)	Actual fitness (3)
Binary	0.995	399.0
Gray	0.995	399.0
Real-value	0.998	400.5

Plots of average fitness in the population with generation number are shown in Fig. 6. Clearly the real-value representation results in a population with a much higher overall fitness than either the binary or Gray code representations. The other significant feature of the real-value representation is that the variability in average fitness between generations is much lower. This is largely because of the improved manner in which mutation can be dealt with. With binary and Gray codings, the average fitness does not improve significantly after about 100 generations. The number of infeasible solutions generated as a result of mutation is higher than with real-value coding, and this restricts the average fitness that can be achieved. In each representation scheme, elitism has been used to ensure that the best individual of a population is not lost between generations, and this ensures that in all representations a high and improving maximum fitness is maintained.

The maximum values of the objective function achieved with 500 generations with each coding are given in Table 1. The known global optimum for the four-reservoir problem is 401.3. Clearly, each of the representation schemes approaches the global optimum closely. Real-value coding appears to offer several advantages over both binary and Gray coding:

- A higher maximum fitness value is achieved
- A smoother convergence to a solution is achieved
- The average fitness of the population generated is much higher

The known global optimum for the four-reservoir problem was reached with the real-value coding representation after 750 generations. Another significant advantage of real-value representation over binary representation is in execution times. On the basis of the parameter sets used above, the real-value representation of the four-reservoir problem completed 500 generations two and a half times faster than the binary representation and four times faster than the Gray coding representation. For larger problems, this speed advantage may be of importance.

Population size also influences the performance of a GA. Fig. 7 shows the influence of population size on maximum fitness produced by the real-value representation after 500 generations. Acceptable results are produced with a population of 100, but by increasing the population size to 200 it is, in fact, possible to match the known global optimum. The total number of evaluations decreases with a decrease in population size, and hence desired results may not be achieved with smaller population sizes. Moreover, the diversity in a population cannot be maintained if the population size is small.

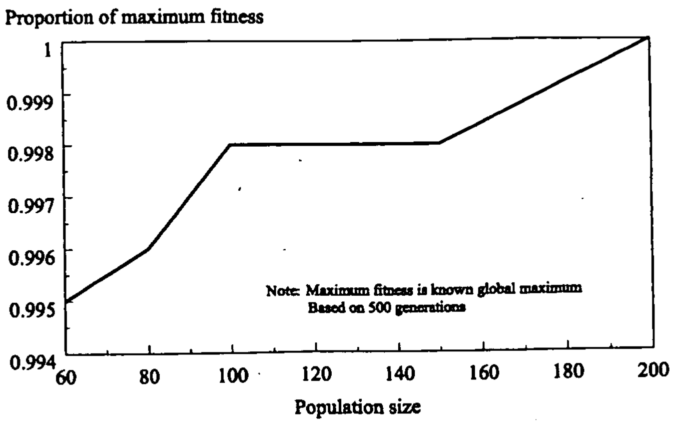


FIG. 7. Influence of Population Size on Fitness

Influence of Alternative Crossover and Mutation Operators

Consideration has been given to the influence of alternative crossover and mutation operators on the results produced with real-value representation. Three crossover and two mutation operators have been considered. Table 2 presents the results with these different approaches. Uniform crossover is the best crossover operator for the four-reservoir problem. This might be expected, as it is likely to lead to greater diversity within the population than either one-point or two-point crossover. Modified uniform mutation is clearly the best mutation operator. With uniform mutation there is a much greater risk of good genetic material being lost. Modified uniform mutation dampens the impacts of mutation, thus leading to better solutions. With modified uniform mutation in real-value coding, the size of mutations can be fixed rather than allowing random mutations to the gene values, and this again leads to better final solutions as disruption to the solution is less. Results are less sensitive to the crossover operator adopted, and uniform and one point crossover produce very similar results.

FOUR-RESERVOIR TRAJECTORIES

A comparison of the trajectories produced by the LP and by the real-value GA for the four-reservoir problem are presented in Fig. 8. The GA results are based on the best parameter set resulting from sensitivity analysis, and are given for a population of 100 at the end of 500 generations. Exact reproduction

TABLE 2. Evaluation of Crossover and Mutation Schemes

Scheme adopted (1)	Proportion of maximum fitness (2)
Crossover Schemes	
Uniform crossover	0.998
One-point crossover	0.997
Two-point crossover	0.994
Mutation Schemes	
Uniform mutation	0.983
Modified uniform mutation	0.998

Note: Modified uniform mutation is used with all crossover schemes, and uniform crossover with all mutation schemes.

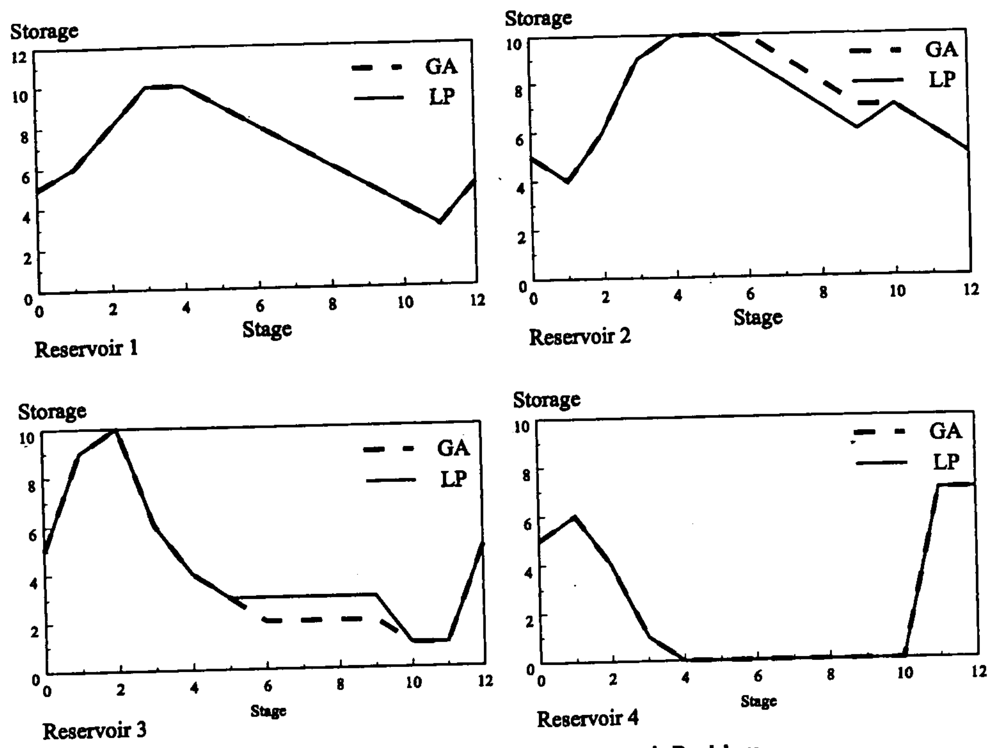


FIG. 8. Trajectories in Four-Reservoir Problem

of LP results was achieved after 750 generations. For reservoir 1, an exact match in trajectories is obtained. In reservoirs 2 and 3, there are minor shifts in trajectories produced with a population of 100. For reservoir 4 there is, with the exception of the first time step, exact reproduction of the optimal trajectory.

The GA also has been run using different seeds in the stochastic generators than had been used in the sensitivity runs. In 5 out of 10 runs with 750 generations, the optimal solution was produced. The other runs produced values that were very close to the optimum (0.999 for three of them and 0.998 for the other two). The results and performance are therefore relatively stable with respect to particular random number sequences.

MODIFIED FOUR-RESERVOIR PROBLEM

GAs have considerable flexibility in application to nonlinear problems. To demonstrate this, the four-reservoir problem has been modified by introducing a head storage relationship for each reservoir of the form $H = K.S^n$, and modifying the objective function to

$$\text{Max } I = \sum_{i=1}^{4} \sum_{t=0}^{11} b_i(t).p_i(t) + \sum_{i=1}^{4} g_i[s_i(12) - d_i] \quad (13)$$

where

$$p_i(t) = f(s_i(t), u_i(t)) \quad (14)$$

is the power generated from reservoir i in time step t.

The modified four-reservoir problem has been solved using discrete differential dynamic programming and using the GA technique. A penalty function is not required for the DDDP technique as it is possible to trace back those trajectories that have the required final state. For the GA the penalty function used with the linear four-reservoir problem has been adopted. The GA was set up with real-value representation, tournament selection, elitism, uniform crossover, and modified uniform mutation. The GA computer code was unchanged from the linear problem with the exception of a modification to the evaluation function. It is perhaps in this flexibility that the main strength of the GA lies.

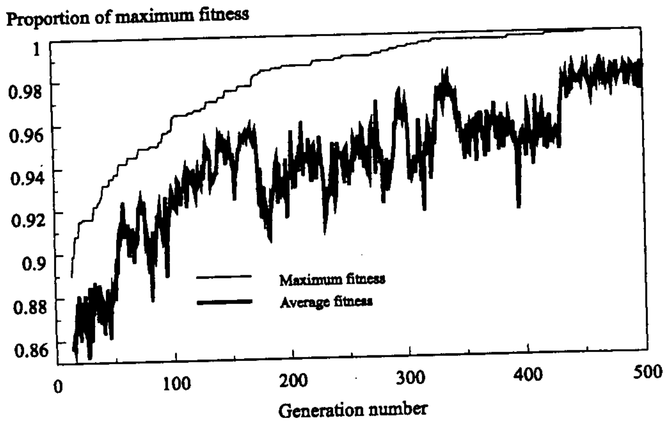


FIG. 9. Fitness Improvement for Nonlinear Four-Reservoir Problem

With the parameters adopted, the optimal return for the problem found using DDDP was 1901.39. With DDDP no improvement was found in this value after 10 iterations. The GA was run with a population size of 100 and achieved maximum fitness after 460 generations. The GA produced the same optimum as the DDDP approach, and trajectories in each reservoir were the same with each technique. Fig. 9 shows the manner in which the GA approached the solution. The average fitness of the population was very high, indicating that a large number of alternative solutions close to the optima have been produced. The GA executed almost three times faster than the DDDP and required no initial trajectories to be set.

APPLICATION TO TEN-RESERVOIR PROBLEM

A ten-reservoir problem was formulated and introduced to the literature by Murray and Yakowitz (1979). The problem is complicated not only in terms of size, but also because of many time-dependent constraints on storage. The problem was formulated such that it remained solvable by linear programning, and Murray and Yakowitz (1979) presented a solution ısing constrained differential dynamic programming. The ›roblem is beyond the capacity of traditional DP and is diffi-:ult with variants such as DDDP, but is relatively simple to ›olve by LP.

The schematic of the ten-reservoir problem is shown in Fig. 10. The system comprises reservoirs in series and in parallel, ınd a reservoir may receive supplies from one or more up-ıtream reservoirs. Operation of the system is to be optimized ›ver 12 operating periods to maximize hydropower produc-ion. Decision variables for the problem are reservoir releases, ınd the state variables are reservoir storages in each operating ›eriod. Inflows are defined for each of the upstream reservoirs, ınd initial storages and target storages at the end of the op-:rating period are specified for each reservoir. In addition,

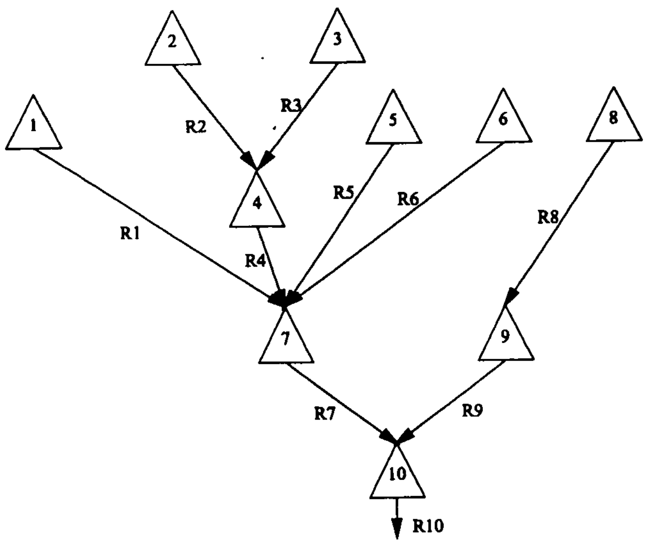


FIG. 10. Arrangement of Reservoirs for 10-Reservoir Problem

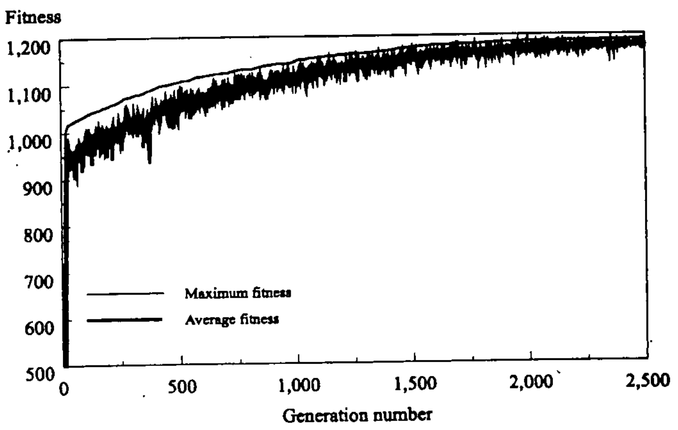


FIG. 11. Maximum and Average Fitness for 10-Reservoir Problem

there are minimum operating storages in each reservoir that must be satisfied, as well as constraints on minimum and maximum reservoir releases. Details are given by Murray and Yakowitz (1979).

The continuity constraints are defined by (7), and the objective function is modified only to reflect the additional reservoirs:

$$\text{Max } I = \sum_{i=1}^{10} \sum_{k=0}^{11} b_i(k).u_i(k) + \sum_{i=1}^{10} g_i[s_i(12) - d_i] \quad (15)$$

Taking the target ending storage in reservoir i to be d_i, the penalty function is expressed as

$$g_i[S_i(12), d_i] = -60[S_i(12) - d_i]^2 \quad \text{for } S_i(12) \le d_i \quad (16)$$

and

$$g_i[S_i(12), d_i] = 0 \quad \text{for } S_i(12) > d_i \quad (17)$$

The penalty function is not required for the LP solution because the target storages can be set up as constraints. It was found that with the ten-reservoir problem a more severe penalty function was required than with the four-reservoir problem.

The components of the solution vectors for the ten-reservoir problem are noninteger continuous values. Discretization within the decision space is not required. The ten-reservoir problem has been solved using the same GA code as used for the four-reservoir problem, with modification only to the evaluation function. The GA parameters found most suitable for the four-reservoir problem were adopted, but the population size was increased to 500 in order to maintain diversity in initial populations. The GA was permitted to run for 2,500 generations and took about 25 minutes to run on a Pentium-based PC. Fig. 11 shows the manner in which convergence to a solution was achieved.

The LP for the ten-reservoir problem was set up in a spreadsheet, and the optimal return of 1,194 obtained. Using constrained DDP, Murray and Yakowitz produced a maximum return of 1,190.652. The maximum return achieved with the GA approach was 1,190.25, which is 99.7% of the known global optima. Given the size of the problem, the results of the GA are very satisfactory. The storage trajectories produced by the GA are very similar to those produced by the LP. The execution time for the GA was eight times longer than that of the LP on a Pentium PC. The execution time of the GA would not, however, have been influenced through the introduction of nonlinear objectives or constraints. This is an advantage for complex systems.

The purpose of this application has been to demonstrate that the GA approach is capable of addressing large problems. Further complexity introduced through nonlinearities in any part of the system would not present any difficulties for the GA approach, and this is a particular strength of the approach.

TABLE 3. GA Performance with Extended Time Horizons

	DDDP	GA, Population of 100			GA, Population of 200		
Stages (1)	Return (2)	Return (3)	% of maximum (4)	Generations (5)	Return (6)	% of maximum (7)	Generations (8)
12	401.3	400.5	99.8	500	401.3	100	500
24	810.6	808.7	99.7	1,300	808.9	99.79	900
36	1,220.1	1,219.1	99.92	2,400	1,218.6	99.89	1,900
48	1,629.6	1,621.2	99.48	2,600	1,626.5	99.81	2,100
60	2,039.1	2,025.1	99.31	2,700	2,036.9	99.89	4,000
72	2,448.6	2,439.2	99.62	4,200	2,446.0	99.89	4,100
84	2,858.1	2,842.5	99.45	4,700	2,847.5	99.63	4,400
96	3,267.6	3,253.3	99.56	5,300	3,259.8	99.76	5,500

APPLICATIONS TO EXTENDED TIME HORIZONS

When the GA approach is applied to problems with extended time horizons, the chromosome length increases, and it may become more difficult to find solutions that satisfy constraints throughout the length of the chromosome. This potential limitation has been investigated by extending the four-reservoir problem to incorporate up to 96 stages instead of the original 12. Results for the problem with extended time horizons produced with the GA has been compared with those produced by DDDP, and are presented in Table 3. In the GA approach, a larger population is required with longer chromosomes in order to maintain diversity in the population. Populations of 100 and 200 have been considered. The number of generations included in Table 3 are those required for the GA to converge to a state at which the change in fitness over 100 generations is less than 0.1 (0.025%). It is clear from Table 3 that the GA maintains high fitness, and that the deterioration in fitness at longer time horizons is not significant. The results indicate that a higher population does produce better overall fitness, although, interestingly, the number of generations required with a larger population is very similar to the number required with a small population. This is another measure of the robustness of the GA approach.

CONCLUSIONS

It has been demonstrated that GAs provide robust and acceptable solutions to the four-reservoir deterministic finite-horizon problem, and can reproduce the known global optimum. Several possible formulations have been considered, along with their sensitivity to various parameters. It is concluded that a real-value representation, incorporating tournament selection, elitism, uniform crossover, and modified uniform mutation will operate most efficiently and produce the best results. A crossover probability of 0.70 is appropriate for the four-reservoir problem, and mutation probability should be based on one mutation per chromosome. For the four-reservoir problem, a solution very close to the known global optimum can be achieved within 500 generations with a population of 100. With a population of 200, the known global optimum can be achieved in 500 generations, and with a population of 100 the known global optimum can be achieved in 750 generations.

The results achieved indicate that there is potential for application of GAs to large finite-horizon multireservoir system problems, where the objective function is complex and other techniques are difficult to apply. A significant advantage of the GA approach is that no initial trial release policy is required, as in discrete differential dynamic programming, for example. The approach is easily applied to nonlinear problems and to complex systems. It has also been shown that acceptable results can be produced over longer time horizons. Furthermore, a GA will generate several solutions that are very close to the optimum, and this gives added flexibility to an operator of a complex reservoir system.

APPENDIX. REFERENCES

Cieniawski, S. E., Eheart, J. W., and Ranjithan, S. (1995). "Using genetic algorithms to solve a multiobjective groundwater monitoring problem." *Water Resour. Res.*, 31(2), 399–409.

Davidson, J. W., and Goulter, I. C. (1995). "Evolution program for the design of rectilinear branched distribution systems." *J. Comp. in Civ. Engrg.*, ASCE, 9(2), 112–121.

Dandy, G. C., Simpson, A. R., and Murphy, L. J. (1996). "An improved genetic algorithm for pipe network optimization." *Water Resour. Res.*, 32(2), 449–458.

De Jong, K. A. (1975). "An analysis of the behaviour of a class of genetic adaptive systems," PhD dissertation, University of Michigan, Ann Arbor, Mich.

Esat, V., and Hall, M. J. (1994). "Water resources system optimisation using genetic algorithms." *Hydroinformatics '94, Proc., 1st Int. Conf. on Hydroinformatics*, Balkema, Rotterdam, The Netherlands, 225–231.

Fahmy, H. S., King, J. P., Wentzel, M. W., and Seton, J. A. (1994). "Economic optimization of river management using genetic algorithms." *Paper No. 943034*, ASAE 1994 Int. Summer Meeting, Am. Soc. of Agricultural Engrs., St. Joseph, Mich.

Franchini, M. (1996). "Use of a genetic algorithm combined with a local search method for the automatic calibration of conceptual rainfall-runoff models." *J. Hydro. Sci.*, 41(1), 21–40.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, Mass.

Goldberg, D. E., and Kuo, C. H. (1987). "Genetic algorithms in pipeline optimization." *J. Comp. in Civ. Engrg.*, ASCE 1(2), 128–141.

Goldberg, D. E., and K. Deb. (1989). "A Comparative analysis of selection schemes used in genetic algorithms." *Foundations of genetic algorithms*, Morgan Kaufman, San Mateo, Calif., 69–93.

Halhal, D., Walters, G. A., Ouazar, D., and Savic, D. A. (1997). "Water network rehabilitation with structured genetic algorithm." *J. Water Resour. Plng. and Mgmt.*, ASCE, 123(3).

Heidari, M., Chow, V. T., Kokotovic, P. V., and Meredith, D. D. (1971). "Discrete differential dynamic programming approach to water resources systems optimization." *Water Resour. Res.*, 7(2), 273–282.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. MIT Press, Cambridge, Mass.

Larson, R. E. (1968). State increment dynamic programming, Elsevier Science, New York.

McKinney, D. C., and Lin, M. D. (1994). "Genetic algorithm solution of groundwater management models." *Water Resour. Res.*, 30(6), 1897–1906.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*. Springer, New York.

Murphy, L. J., Simpson, A. R., and Dandy, G. C. (1993). "Design of a network using genetic algorithms." *Water*, 20, 40–42.

Murray, D. M., and Yakowitz, S. (1979). "Constrained differential dynamic programming and its application to multireservoir control." *Water Resour. Res.*, 15(5), 1017–1027.

Oliveira, R., and Loucks, D. P. (1997). "Operating rules for multireservoir systems." *Water Resour. Res.*, 33(4), 839–852.

Ritzel, B. J., Eheart, J. W., and Ranjithan, S. (1994). "Using genetic algorithms to solve a multiple objective groundwater pollution problem." *Water Resour. Res.*, 30(5), 1589–1603.

Savic, D. A., and Walters, G. A. (1997). "Genetic algorithms for least-cost design of water distribution networks." *J. Water Resour. Plng. and Mgmt.*, ASCE, 123(2), 67–77.

Schaffer, J. D., Eshelman, L. J., and Offutt, D. (1989). "Spurious correlations and premature convergence in genetic algorithms." *Foundations of genetic algorithms*, Morgan Kaufman, San Mateo, Calif., 102–111.

Simpson, A. R., Dandy, G. C., and Murphy, L. J. (1994). "Genetic algorithms compared with other techniques for pipe optimization." *J. Water Resour. Plng. and Mgmt.*, ASCE, 120(4), 423–443.

Wang, Q. J. (1991). "The genetic algorithm and its application to calibrating conceptual rainfall-runoff models." *Water Resour. Res.*, 27(9), 2467–2471.

Whitley, D. (1989). "The GENITOR algorithm and selection pressure: Why rank best allocation of reproductive trials is best." *Proc., 3rd Int. Conf. on Genetic Algorithms*, J. D. Schaffer, ed., Morgan Kaufman, San Francisco.

Yang, J. P., and Soh, C. K. (1997). "Structural optimization by genetic algorithms with tournament selection." *J. Comp. in Civ. Engrg.*, ASCE, 11(3), 195–200.