A STRONGLY FEASIBLE EVOLUTION PROGRAM FOR NON-LINEAR OPTIMIZATION OF NETWORK FLOWS

Ph.D. Thesis Submitted to:

UNIVERSITY OF MANITOBA
DEPARTMENT OF CIVIL AND GEOLOGICAL ENGINEERING

Submitted by:

Nesa Ilich
ID 6725635

September 2000

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

A Strongly Feasible Evolution Program for Non-Linear Optimization of Network Flows

BY

Nesa Ilich

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of Doctor of Philosophy

ABSTRACT

This thesis describes the main features of a *Strongly Feasible Evolution Program (SFEP)* for solving network flow programs that can be non-linear both in the constraints and in the objective function. The approach is a hybrid of a network flow algorithm and an evolution program. Network flow theory is used to help conduct the search exclusively within the feasible region, while progress towards optimal points in the search space is achieved using evolution programming mechanisms such as recombination and mutation. The solution procedure is based on a recombination operator in which all parents in a small mating pool have equal chance of contributing their genetic material to an offspring. When an offspring is created with better fitness value than that of the worst parent, the worst parent is discarded from the mating pool while the offspring is placed in it. The main contributions are in the *massive parallel initialization* procedure which creates only feasible solutions with simple heuristic rules that increase chances of creating solutions with good fitness values for the initial mating pool, and the *gene therapy procedure* which fixes "defective genes" ensuring that the offspring resulting from recombination is always feasible. Both procedures utilize the properties of network flows. Tests were conducted on a number of previously published transportation problems with 49 and 100 decision variables, and on two problems involving water resources networks with complex non-linear constraints with up to 1500 variables. Convergence to equal or better solutions was achieved with often less than one tenth of the previous computational efforts.

Key Words: Genetic Algorithms, Evolution Programs, Network Flows, Non-Linear Constraints

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The objective of this research is to develop an algorithm for solving minimum cost network flow problems associated with mixed-integer decision variables (flows) and non-linear objective function and flow constraints. There are currently no known specialized network solvers for this problem. The research will focus on the ability to find the global optimum, rather than the execution efficiency, which may only be addressed in the final stages of this research.

The field of *network optimization* has evolved as an important branch of operational research in the last few decades with application in many areas of engineering and management. The need for this research was inspired by two possible engineering applications -- one related to water allocation in complex river basin networks and the other related to optimizing pipeline operation. Both problems have non-linear constraints and objective function. The available solution procedures that have been used in the past for solving these types of problems had limited success. They were addressed using linear programming (LP) solvers applied to problems which were formulated in terms of separable cost functions and linear constraints. The objective is to find the best possible allocation within a given time period in complex river basin networks. To justify the use of linear constraints, the problem had to be solved using iterations and large calculation time steps. Such solutions were often of limited practical value for real-time operation. Consequently, the use of the existing models with linear programming solvers has so far been restricted to planning studies.

The field of optimizing pipeline operation has been another important area of interest to researchers. The principal goal is to deliver the target volume through a pipeline within a given time period such that the cost of pumping is minimized. While the pressure constraints are non-linear functions of flow, the problem also has difficult non-linear cost functions. It can also involve mixed-integer decision variables, resulting from the fact that some of the pumps operate with a fixed speed while others can operate with a variable speed. Very few

solvers are available to address this kind of problem. The GAMS library of solvers, which is considered as the standard in the operational research community nowadays, offers more than twenty available modules, but only two of them are capable of addressing the class of mixed-integer non-linear problems. It comes with a disclaimer that it is only capable of finding local optimums since it relies on a gradient search approach.

The limitations of various existing search methods to find global optimums for complex constrained problems with objective functions that had numerous local optimums have inspired researchers to look for other solution techniques. This gave rise to the recent development of the search methods known as *Evolution Programs* (Michailewicz 1994), which have become popular in the last two decades due to their effectiveness and have also inspired many researchers to focus their attention in this field. While there is no universal evolution programming solver applicable to all methods, this research will attempt to show that certain properties of network flows can be effectively utilized in the search procedure. The proposed methodology combines the knowledge of network flow theory with the recent achievements in evolution programming. The following is the expected contribution of the proposed research:

- A generalized approach to solving non-linear network flow problems which could be applicable to various fields of engineering, although the testing may be focussed on problems of interest to water resources;
- The proposed algorithm would allow cost functions of arbitrary shape for each individual arc in the network, which cannot be handled by most frequently used solutions methods which rely on the gradient search approach;
- The proposed algorithm would allow linear or non-linear relationships between arc flows in the network, both in terms of loss or gain of flow along an arc, as well as in terms of the mutual relationship between flows on two or more arcs; and,
- The method would also be able to handle mixed integer programs with non-linear objective functions or constraints.

The scope of this research includes the following:

- review of recent developments in evolutionary programming as well as a review of pertinent background from the network flow theory;
- formulation of an algorithm which combines the building blocks of flow defined in the flow network theory as the basis for the proposed evolutionary search;
- development of the computer code for a generalized network solver;
- testing of the code by comparing its performance with other published test results;
- testing of the code on new challenging problems that could not have been solved with other solvers; and,
- publishing the results of the above tests in selected engineering journals or conference proceedings.

This document starts with introduction of the basic concepts of networks and theory of network flows, followed by a review of historic developments in network flow optimization. The summary of historic developments includes a review of strengths and weaknesses of earlier linear and non-linear solution techniques, which justifies the need for further research. One section is devoted to introducing the reader to the main application area of interest -- water resources networks. This application is converted into network flow problems and the nature of the constraints and the objective function is discussed. Chapters 4, 5 and 6 include a review of the relevant properties of network flows which are used in the process of building a solution technique, a review of evolutionary programming, and a review of the proposed solution technique, respectively. Finally, chapters 7, 8 and 9 include the three case studies where the new solver has been applied.

2 BACKGROUND AND LITERATURE REVIEW

Because of their widespread applicability in various fields, networks are one of the most extensively studied topics in the last few decades. A bibliography entitled "Deterministic Network Optimization" compiled over twenty years ago had more than 1000 entries, and that was still only a partial list (Golden and Magnanti, 1977). In general, networks are studied in terms of their structure or in terms of associated functions, hence the problems of *network synthesis* or *network analysis*, although for many real world problems it is sometimes difficult to keep a clear distinction between the two. Of primary interest to this research is analyses of single commodity flows in networks (Ford and Fulkerson, 1962).

The following sections provide introduction to basic concepts and theorems of network flows from the graph theory. These concepts form the basis of the proposed algorithm.

2.1 Definitions and Review of Related Terminology

The theory of graphs is a large body of mathematics with many topics on networks and network flows (Busacker and Saaty, 1965; Rockafellar, 1984; Ahuja et al., 1993). In spite of the volumes of theoretical contributions from various researchers, theory of graphs is still missing universal notation. Each textbook or publication on networks starts with defining its own notation. This review follows the notation used in *Network Flows* (Ahuja et al., 1993) as one of the most recent and the most relevant references in this field.

There is no clear distinction between networks and graphs in the literature. Some researchers make a distinction by defining networks as special types of graphs which have a flow function associated with them. In most general terms, a graph is defined as a non empty set of nodes N, a possibly empty set of arcs A and a mapping function E between N and A (Busacker and Saaty, 1965). Nodes are also called vertices or points, arcs are also known as links, edges or branches, and the mapping function is usually referred to as the incidence

mapping. Networks (as special types of graphs) can therefore be defined by using G = (N, A, E) or simply by (N,A) which implicitly includes the incidence mapping E. The incidence context is fundamental to a graph, and the usual notation for an arc is (i,j) which means that the arc is incident with nodes i and j, or that its end points are i and j. A few more terms are used in the subsequent sections of this document and they are defined below.

Directed (or oriented) graphs differ from undirected graphs in the property that elements of set A (arcs) are defined as ordered pairs of distinct nodes, as opposed to undirected graphs where ordering is not required. In terms of networks, arc orientation allows flow in only one direction. This does not pose a limitation in mathematical programming, since algorithms which require only non-negative decision variables can use two non-negative decision variables x' and x" which are related to the original decision variable in the form of:

$$x' - x'' = x \qquad (2.1)$$

where x is unrestricted in sign. In the network, this transformation is equivalent to splitting an undirected arc into two parallel directed arcs with opposite orientation which have non-negative flows x' and x" associated with them. A directed arc (i,j) has two endpoints, i and j usually referred to as the *head node* and the *tail node*.

Loops (arcs with the same head node and tail node, i.e arcs which originate and terminate at the same node) have very little use in network flow analysis and they will not be considered in the following. *Multiarcs* (several parallel arcs which all share the same tail node and the same head node) will also be excluded from further analysis with one important qualifier: except for notation (i,j) which allows the existence of only one arc with the tail node i and head node j, all other rules and algorithms discussed in the following apply equally to networks with multiple arcs.

Nodes in the network are classified as *sources*, *sinks* or *transshipment nodes*, depending on their respective positive, negative or zero balance of inflows and outflows. *Circulatory* networks are those which contain only transshipment nodes and their flows are called

circulations, since flow does not leave or enter the network at any point. The problem of finding optimal circulation is equivalent to the problem of finding optimal flows, since any standard network can be converted to circulatory by adding one additional node to it (usually termed *universal source/sink* or *system balance* node), and by adding additional arcs oriented from this node to all other sources as well as additional arcs oriented from the sink nodes to the system balance node.

A *walk* in a directed graph G=(N,A) is a subgraph G' which consists of a subset of mutually adjacent nodes and arcs from G. Because of the existing incidence relationship between nodes and arcs walk can be defined only as a subset of nodes (a subset of arcs). A *directed walk* is a walk which consists of arcs which have the same orientation.

Path is a walk without any repetition of nodes. Arcs which belong to a path are classified as *forward* or *backward*, depending on their orientation. A *directed path* is a directed walk without any repetition of nodes, in other words a path without any backward arcs. Path in a network is a walk from the source to the sink.

Cycle is a path which begins and ends in the same node. A *directed cycle* is a cycle which consists of arcs oriented in the same direction.

Nodes i and j are *connected* if the graph contains at least one path from node i to node j. A network is connected if every pair of its nodes is connected, otherwise it is *disconnected.* Strong connectivity implies existence of at least one directed path from each node to every other node in the network.

A *cut* is a partition of set of all nodes N into two subsets, S and N \ S. Finally, a *tree* is a connected graph which contains no cycle. *Maximum spanning tree* or *maximum forest* is a connected graph which contains all nodes N of network G while it contains no cycle.

2.2 Network Flows

Network flow can be defined as a function (or a vector) which associates a value x_{ij} with every arc in the network. The *minimum cost flow* problem is the most general network flow problem. Prior to introducing a mathematical formulation of the minimum cost flow problem, the concepts of *bounded* networks and *feasible* flows will be established. Network is bounded if functions l_{ij} and u_{ij} are associated with every arc (i,j) in the network such that:

$$0 \le l_{ij} \le u_{ij} \qquad \forall (i,j) \in A \tag{2.2}$$

Functions l_{ij} and u_{ij} are termed the lower bound and the upper bound imposed on flow on arc (i,j). When $l_{ij} = 0$ and $u_{ij} > 0$ the network is termed *capacitated*. A circulation consisting of a set of arc flows x_{ij} is feasible if it satisfies the following conditions:

$$\sum_{\{j(i,j)\in A\}} x_{ij} - \sum_{\{j(j,i)\in A\}} x_{ji} = 0 \qquad \forall i \in N \tag{2.3}$$

$$0 \le l_{ij} \le x_{ij} \le u_{ij} \qquad \forall (i,j) \in A \tag{2.4}$$

The first expression is a matrix equation with A columns and N rows showing that summation of all arc flows incident to a given node equals zero in a circulation. The second condition forces the flows on each arc to comply with the bounds. Finding vector x_{ij} which satisfies the above conditions constitutes the problem of finding a feasible circulation. Properties of feasible circulations are of significant importance to the developments of the ideas in this research and they will be addressed in more detail in Chapter 4.

One more vector is required to define the minimum cost flow problem for a network, known as the arc cost c_{ij}. It associates a cost of sending a unit of flow from node i to node j along arc (i,j). The minimum cost flow (or circulation) is then defined as the problem of finding a feasible circulation x_{ij} which also minimizes the total cost of flow in the network. This can be mathematically expressed as:

$$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij} \qquad \forall (i,j) \in A \tag{2.5}$$

subject to the feasibility constraints (2.3) and (2.4). Hence, *optimal circulation* is the one which minimizes the total cost of flow in the network while satisfying the feasibility constraints. There can be one or more circulations which are termed optimal.

If functions l_{ij}, u_{ij} and c_{ij} are arrays of constant parameters, the above is a linear program. Many solution procedures are available for solving the above problem if it can be represented as a linear program. However, if parameters l_{ij}, u_{ij} and c_{ij} are not constant even for only one arc in the network, the above problem becomes a non-linear program which is much harder to solve. In some cases non-linearities can be 'linearized' and the problem can be converted to an approximate linear program. In other cases this cannot be done and the program must be solved using non-linear programming techniques. The benefits and down sides of both approaches are summarized in the following. Much of the algorithmic development has been done in the area of linear programming, while the developments in non-linear programming lag behind to some extent due to larger complexity and smaller theoretical foundation in comparison to linear programming. Water resources networks (which represent river basins and the accompanying set of irrigation canals, reservoirs and other components) are characterized with extreme non-linearities and sizeable network complexities, yet most of the applications to date have relied on linear programming approach with various approximations and simplifications.

2.3 A Summary of LP Applications to Network Flow Problems

A typical textbook on network programming has almost 90% of its contents devoted to linear programming applications. There is a large body of available algorithms and theoretical developments, which originated in the 1950s (Dantzig, 1963; Ford and Fulkerson, 1962). Normally, an LP program is defined in terms of minimization (or maximization) of an objective function which has a linear form, subject to a set of inequalities. The program defined by expressions (2.3) - (2.5) could also be represented in this format, by converting each equality in Expression (2.3) into a set of two inequalities using the general rule that each

equation of the form a = b is equivalent to a set of two inequalities a ≤ b and a ≥ b. Re-writing the program (2.3) - (2.5) using this transformation would provide a definition of the minimum cost flow problem as a linear program in its canonical form, suitable for Simplex and other popular LP solvers. However, networks have some special properties which inspired researchers to develop much more efficient algorithms. Instead of converting the equalities into inequalities, these algorithms use the equalities in expression (2.3) and take advantage of them. The cycle cancelling algorithm, the successive shortest path algorithm, the out-of-kilter algorithm and the network simplex algorithm and their variants are all examples of these developments. They all take advantage of the equality constraints (2.3). The ultimate application of network simplex algorithm was extended to problem of optimizing generalized flows, which can mathematically be expressed as:

$$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij} \qquad \forall (i,j) \in A \qquad (2.6)$$

subject to:

$$\sum_{\{j(i,j)\in A\}} x_{ij} - \sum_{\{j(j,i)\in A\}} \mu_{ij} x_{ji} = b(i) \qquad \forall i \in N \qquad (2.7)$$

$$0 \le l_{ij} \le x_{ij} \le u_{ij} \qquad \forall (i,j) \in A \qquad (2.8)$$

The new term μ_{ij} is called the *arc multiplier* of arc (i,j) and it is a rational number. If $\mu_i = 1$ for all arcs (i,j) the problem is similar to the one previously defined by expressions (2.3) - (2.5), however if $0 < \mu_{ij} < 1$ then there is a loss of flow along arc (i,j) while if $\mu_{ij} > 1$ arc (i,j) gains flow from node i to node j. Note that the gain and loss functions must be linear to allow application of generalized network simplex algorithm, while gain or loss of flow in real world problems may not follow a linear function. An LP program defined by expressions (2.6) through (2.8) is no longer a circulation, as can be seen from the right hand side of expression (2.7) which no longer equals zero. The value of each b(i) defines a source if b(i) > 0, sink if b(i) < 0 and transshipment node if b(i) = 0, while c_{ij} is the cost per unit flow entering the arc at node i. Generalized network flow problems are usually more difficult to solve, although some recent developments of new solvers claim significant improvements

in execution speed. There is a vast body of literature on the algorithms for minimum cost flow problems and for generalized network flows. The following literature review is attempted to capture the most significant works in this field.

2.4 Historic Developments in Linear Network Optimization

The following is a list of major achievements related to the development of algorithms for solving the linear minimum cost flow problem in constrained networks. Ford and Fulkerson initially developed primal-dual algorithms for transportation problems. They later generalized this approach for solving the minimum cost flow problem (1962). Jewel (1958), Iri (1960) and Busaker and Gowen (1961) independently developed the shortest path algorithm and showed how to solve the minimum cost flow problem as a sequence of shortest path problems. Fulkerson (1961) and Minty (1960) have independently developed the out-of-kilter algorithm, a specialized network solver for minimum cost flow problems which consists of a sequence of changing primal and dual variables such that the optimality conditions derived from the complementary slackness theorem are eventually reached. Klein (1967) developed the cycle cancelling algorithm which maintains feasibility at every step as it tries to converge to optimality, as opposed to the successive shortest path algorithm which maintains the non-negativity cycle costs and flow capacity constraints, but violates the mass balance constraints at the nodes. Further improvements of cycle cancelling algorithm are due to Barahona and Tardos (1989) which modified the algorithm of Weintraub (1974), Goldberg and Tarjan (1988), and Wallacher and Zimmerman (1991) which all use a different choice of augmenting cycles to improve convergence efficiency. Zadeh (1973) provided a comparison of efficiency of cycle cancelling, successive shortest path and out-of-kilter algorithms. Edmonds and Karp (1972) introduced the scaling approach for the minimum cost flow problem, based on the capacity scaling technique. Rock (1980), Orlin (1988), and Bland and Jensen (1992) all experimented with a scaling technique for the minimum cost flow problem. Goldberg and Tarjan developed several improved implementations of the ϵ-optimality concept, which was independently suggested by Bertsekas (1979). Goldberg and

Tarjan (1987), Ahuja, Goldberg, Orlin and Tarjan (1992) have also developed specialized applications for minimum cost flow problem.

Comparisons of computational efficiency of various algorithms were conducted by Barr, Glover and Klingman (1974); and Bradely, Brown and Graves (1977). They concluded that the best algorithms are the network simplex algorithm and the relaxation algorithm, developed by Bertekas and Tseng (1988). Kenington and Helgason (1980), Jensen and Barnes (1980) and Ahuja, Magnanti and Orlin (1993) give substantial treatment of the generalized network simplex algorithm in their textbooks, although in different ways. Elam, Glover and Klingman (1979), Brown and McBride (1984) have examined computational performance of the generalized network simplex algorithm, which is believed to be the fastest available algorithm for solving the generalized network flow problem in practice. Brown, McBride and Wood (1985) have created EMNET program which solves combined generalized network problem with additional non-network linear constraints. Sun et al. (1995) provide details of EMNET application and report on its computational efficiency.

2.5 Linear Programming Limitations

The most significant limitation of linear programming is the assumption that all constraints (upper bounds, lower bounds and costs) can be approximated as linear functions of flow. This is often not the case in water resources networks, as will be discussed at length in the next chapter. However, in many cases non-linear programs can be 'linearized' and solved using the existing LP solvers.

There is one more aspect of linear programming which can sometimes complicate its use, associated with the decision variables of equal priority (cost). Consider for example a problem of finding a minimum cost flow in a network with two or more arcs whose costs are equal. The value of the objective function will then be the same for various combinations of flows in those arcs which yield the same overall flow in the network (the sum of all flows

in the network). This has the unfortunate consequence of the existence of more than one solution with the same optimality. In practical terms, this may prevent computer models from finding a unique solution, and every small and immaterial change in the input data file for repeated simulation runs may result in a different solution which is equally optimal. While this may not by considered as much of a technical problem, it is not socially acceptable in many instances where modelling is subjected to public scrutiny, as is the case for example in the water resources field. The problem of finding an equitable distribution for a set of variables often arises in complex water resources networks, where for example several water users of the same type are to receive water with equal priority. During shortages, an LP model may cut supply to some users completely while the others are still receiving their target levels. In practical terms, the LP model has failed to deliver equitable supply to all users of equal priority. To avoid this problem, it is necessary to rank the users in the same type and allow controlled differences. This means that if the number of users in the same group is k and the maximum allowed difference in deficit relative to their target is 1 percent, it is necessary to split each decision variable into 100 new variables each representing 1 percent of the target demand, so the number of variables is increased from k to $100k$ and each of them must be assigned a unique cost c_{ij}. In terms of network reformulation this is equivalent to splitting each arc representing a user from the given group into 100 parallel arcs. It is a workable option if 1 percent differences can be tolerated, but still messy and very inefficient. Yet in other instances 1 percent differences may constitute significant violation of equal supply thus further restricting the use of LP.

2.6 Non Linear Network Optimization

While there are special types of solvers for specific linear programs, every linear program can be solved using the universal *Simplex* algorithm (Danzig, 1963). Such universal algorithm for non-linear problems does not exist (Hiller and Liberman, 1995; Avriel, 1976). In general, non-linear programs are much more difficult to solve due to the following:

- optimal point in non-linear program may not be a corner point of a feasible region. Instead, it can be any interior point within a feasible region, which creates a significantly larger search space in comparison to linear programs where only the corner points need to be examined; and,

- except in a few special cases, non-linear programming algorithms are unable to distinguish between a local minimum and a global minimum (except by perhaps finding all local minimums). In many complex problems there is no mathematical proof to guarantee the existence of a global optimum.

In spite of these drawbacks, many special cases related to the constraints and the objective function have been addressed successfully with specialized algorithms. The following are some of the better known non-linear programming algorithms. Their strengths and weaknesses regarding possible applicability to water resources networks are reviewed in Chapter 3.

The only class of Non-LP programs which can guarantee that local minimum is also a global minimum are known as the *convex programming* problems, which have convex objective function while constraints are all described using concave functions (Kuhn and Tucker, 1951; McCormick, 1983). It will be seen that water resources networks described in Chapter 3 fail both criteria. Hence, the existing Non-LP algorithms cannot guarantee finding a global minimum for either one of the two problems. Some special cases of Non-LP algorithms that have been applied to water resources networks are outlined in the following.

2.7 Linearly Constrained Programs

There is a large class of Non-LP programs which have linear constraints and non-linear objective function. They include *quadratic programs* if the objective function is quadratic, which is a sub-set of a larger class of convex problems described by an objective function

that is concave and constraints that are convex. Of particular interest is a special case of convex programming where one additional assumption is valid, i.e. where the constraints and the objective are represented by *separable functions*, which means that they can be broken down into a finite number of individual linear functions. Separable programming (Danzig, 1963) was used in river basin allocation models mentioned in section 3.1.5 with a restriction that the decision time step be long such that the non-linearities related to river channel routing can be approximated with linear functions. Additional linearization of non-linear functions related to hydro power, reservoir and weir outflows is typically handled using an iterative process built into the model. Limitations and possible errors related to using iterative procedures are demonstrated in Chapter 3.

2.8 Other Non-Linear Search Methods

There is a large number of non-linear optimization algorithms which fall into the category of *noncovex programming* and they are usually much more difficult to solve. As mentioned earlier, most of them were developed for a specific class of problems. At this point, there seems to be no specialized algorithm for solving non-linear network flow problems for any type of decision variables and non-linear arc bounds. The most general classification of non-linear search methods is on *direct search* which require only the objective function values and *gradient search* methods which require estimates of the partial derivatives. The oldest direct search method is known under a variety of names (parallel axis method, univariate search, etc.) has been attempted by many researchers in a large number of variations. The basic idea was to fix all coordinates except for one, which is varied in the direction of the axis by a small positive and negative change. The point with a better value of the objective function thus becomes the starting point for the evaluation with respect to another variable, which is evaluated parallel to its coordinate axis. The search progresses in this fashion until no further improvement in the value of the objective function can be found (Kowalik and Osborne, 1968; Schechter, 1968; Ortega and Rheiboldt, 1967). Improvements of this strategy were due to Hooke and Jeeves (1961) who introduced the direct pattern search steps

which were not parallel to the coordinate axis, as well as Rosebrock (1960) who introduced a strategy of rotating coordinates. Both improvements were aimed at reducing the limitation on the number of search directions. A combination of Rosenbrock's ideas were expanded by Swan (1964) and later by Box, Davies and Swan (1969). Nelder and Mead (1965) proposed the so called 'simplex' search strategy (which has nothing to do with simplex method of linear programming). This strategy evaluates the objective function at $n+1$ corners of the polyhedron and moves in the most promising direction accordingly, where n is the number of variables or the dimension of the search space.

In gradient search the partial derivatives of the objective function are evaluated at each step of the search in order to determine the best search direction for the next step. Kantorovich (1945), Levenberg (1944) and Curry (1944) are considered the originators of the gradient strategy. A variant of this strategy is known as the steepest descent (for maximization: ascent) method (Brown, 1959). Other authors who investigated this strategy and its convergence include Goldstein (1962), Ostrowski (1967) and Wolfe (1969, 1970, 1971).

The gradient strategy is of local character and it cannot distinguish between the global and local optimums. To increase chances of finding a global optimum, it is necessary to start the search frequently from various initial values of the decision variables and compare all optimums found in this repetitive process (Jacoby, Kowalik and Pizzo, 1972). The method of conjugate directions (Powel, 1962; Fletcher and Reeves, 1964) aims to speed up the search by evaluating the second order partial derivatives. These methods use the values of the objective function gathered in the search process to estimate the values of the partial derivatives numerically, and as such they are considered Quasi-Newton methods. They differ from the pure Newton strategy (Householder, 1953; Goldstein, 1965) which requires no explicit values of the objective function, but it does require evaluation of the first and second order derivatives, a task that often requires a considerable effort. Very few algorithms can determine the first and the second order partial derivatives numerically using the trial and error approach (Wegge, 1966). The approximation errors that accumulate in the process

often cancel out the advantages of the method which are apparent when the partial derivatives are known. Efforts were focussed on ways to estimate second partial derivatives based on the values of the objective function obtained in the previous steps (Brown and Dennis, 1972; Gill and Murray, 1972; Fletcher and Powel 1963).

It is recognized that often times the objective function may be too complex for derivation or it simply may not have derivatives for all values of the decision variables. Since all of these methods move through the feasible region in a step-by-step fashion, it is conceivable that their chances of finding a global optimum in problems which are riddled with local optima are small. The efforts in this research will exploit a new generation of evolutionary search methods (Michalewicz, 1994), which tackle the search process from all directions within a feasible region. Even though there is no theoretical proof that they always converge to a global optimum, the current state of the art confirms that they are capable of finding much better local optima than the standard methods, and also that they have the capability to find the global optimum in many problems, given the appropriate setup of convergence parameters. They are addressed in Chapter 5 in more detail.

2.9 Dynamic Programming

Dynamic Programming (DP) offers possible advantages over other search methods since it is not affected by the shape of the objective function. DP requires discretization of the problem into a finite set of stages in the search process. At every stage a number of possible conditions of the system (states) are identified and an optimal policy is identified at each individual stage given that optimal policy for the next stage is available (Belman 1957; Bersekas, 1987; Sniedovich, 1991). In short, the main features of DP approach are:

- DP approach works with a finite number of states (possible outcomes). Therefore, the accuracy of the solution is defined by the initial discretization of the problem.
- DP applications are not general, each is developed for a specific problem and if

something is changed in the configuration of the problem the program coding must also be changed and tested. The entire process of developing a DP application requires considerable experience and judgment, and it is usually problem specific rather than general.

- DP applications are usually more computationally expensive than other methods due to slower execution.

In closing, most researchers have been looking for new approaches which would combine efficiency and ability to find the global optimum. *Evolutionary Programming* approach is proposed in this research as it seems to hold out a promise to achieve both. As a topic of special interest it is reviewed in a separate chapter.

3 WATER RESOURCES NETWORKS

Constraints related to water supply are associated with existing physical paths, such as canals, rivers, pipelines, as well as control structures which regulate the flows such as dams, weirs or pressure valves in pipelines. The problems of water supply can be viewed as network programming, since the upper bounds, lower bounds and arc costs have a tangible real world representation in water supply networks. They represent design canal flow limits or reservoir storage capacities.

3.1 Network Representation of River Basins

A central unit of water resources analysis for any region is the river basin, which comprises all natural watercourses and man-made structures within the boundary of a given watershed. Nodes in a river basin network represent the locations in the river where flow is joined or split, such as a confluence or a weir. These can be represented as the transhipment nodes. Reservoirs are also nodes in river basin networks, and they can act either as sources or as sinks, depending on their current mode of operation (refill or release). Hence, in a circulatory network representation at least two arcs with opposite orientation are added to the system, connecting the reservoir with the system balance node. The sum of flows in these two arcs represents reservoir release or refill, depending on its sign. Source nodes represent locations on the boundary of the modelled region where water is made available to the system (inflows), and also at the locations where water is lost to the system (e.g. evapo-transpiration from irrigated land or other consumptive use). Figure 3.1 shows a schematic representation of a water resources system with typical components which are discussed below. Modelled region is a part of the basin being studied, and it can include the entire watershed or parts of it, according to the desired objectives.

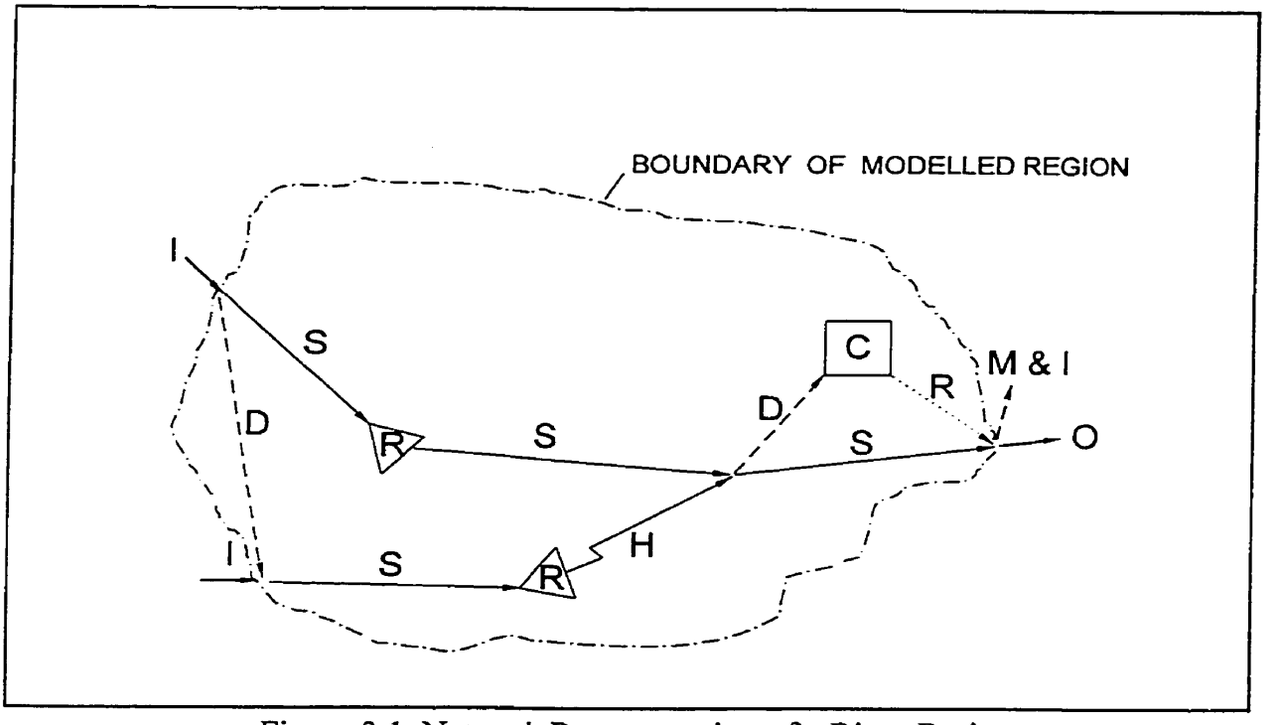


Figure 3.1 Network Representation of a River Basin

Symbols used in Figure 3.1 denote the following:

I	inflow (flow in the river or canal on the boundary of the modelled region);
S	stream (river segment between the two nodes);
R	reservoir ;
D	diversion canal;
C	consumptive use node (location with significant losses of water to the system such as evapotranspiration on irrigated blocks);
R	return flow channel;
H	hydro power channel;
M&I	municipal and industrial diversion; and,
O	outflow from the system at the point where the modelled region ends.

Before each of the above components is discussed, it should be mentioned that optimal flow calculations in water resources networks are associated with one or more incremental time steps, and that estimates of all demands in the system, including crop demand on irrigated

land and hydro power demand for all hydro power plants in the system should be available and converted to the equivalent water requirements (volumetric or flow-equivalent) within a given time step. Hence, the following review is restricted to the deterministic approach. Figure 3.2 shows a circulatory network corresponding to the system depicted in Figure 3.1. The new node added to the system is node B (system balance node). The flow in arcs oriented from node B to a source node in the system represent inflows into the system, while the arcs oriented towards node B represent losses (flows out of the system). Increase of storage in reservoirs is considered as a loss within a given time interval, while reduction of storage amounts to extra inflow for a given time step. Each of the arcs in Figure 3.2 has the upper bound, lower bound and a cost function representing the priority of allocating flow to it. The cost function can be associated with economic cost-benefit analysis, representing the cost of deficit that each water user would suffer for various range of shortages. This function is often difficult to evaluate, since it is hard to attach a dollar value for lost fisheries or recreational and aesthetic aspects of low flows in natural streams. The conflicting interests of consumptive (industrial and agricultural users) and in-stream users (other social groups representing environmental and other water use concerns) are resolved through a political process for which no final formula exists. Certain reasonable assumptions can be made in general to address this issue. It will be assumed in the following that a sharing policy between all components exists (including possible equal sharing as a subset of the entire policy). The primary goal here is to show the extreme degree of non-linearity of the arc flows and bounds.

Inflow nodes in Figure 3.2 are labelled with I, reservoirs with R and the consumptive use nodes with C. There is one node without any label in Figure 3.2, and this is a simple transshipment node. Given arc bounds and costs, the problem of optimal water allocation in the basin can be stated using expressions (2.5), (2.4) and (2.3) or in more general terms using expressions (2.6) - (2.8). The following is a discussion related to the arc bounds for each of the component types depicted in Figure 3.2. Inflow arcs have their upper and lower bound set equal to the value of flow entering the modelled region. They represent 'hard

constraints' in this manner, i.e they impose a certain inflow into the system.

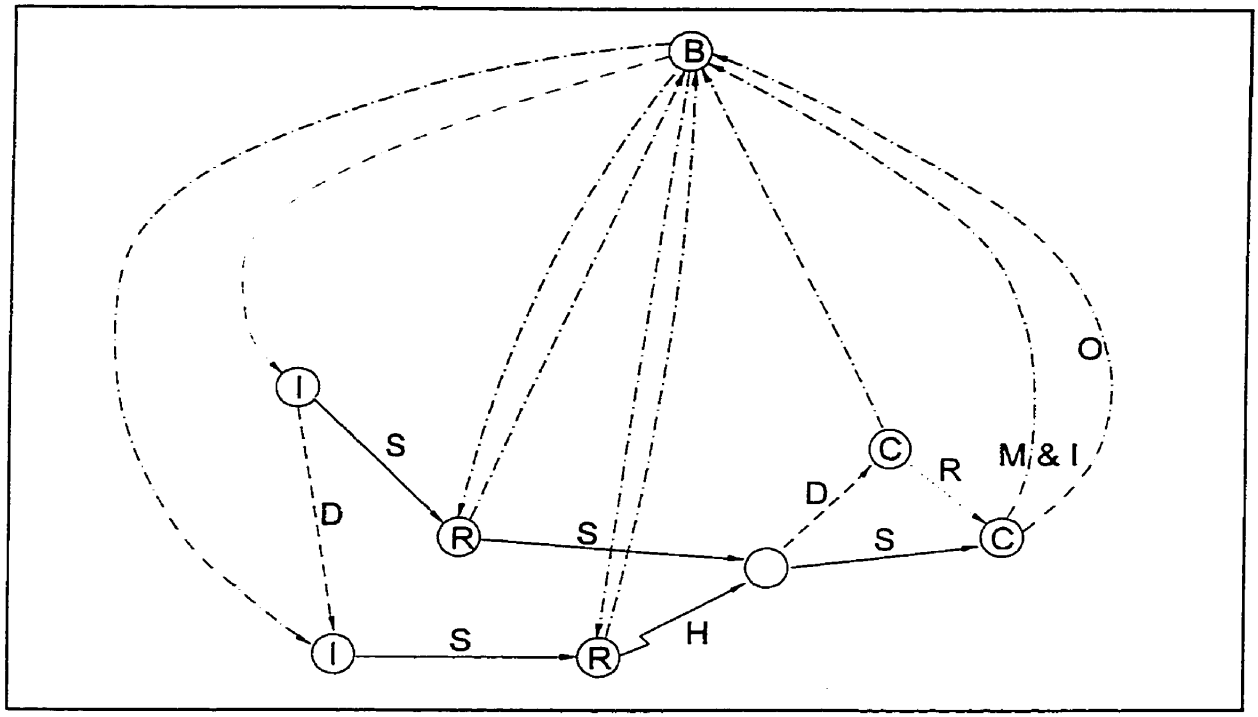


Figure 3.2 Circulatory Network Representation of a River Basin

3.1.1 Flow Conveyance Constraints

An arc representing a natural channel that does not have a reservoir as its upstream node has the lower bound equal to zero and the upper bound equal to the maximum flow that can possibly be routed through a channel, which is normally a large user-defined number. The difficulties exist in the differences between the inflow into the channel at its tail node and the outflow at its head node, which can be caused by several factors:

- for short time intervals and longer river segments, the channel routing effects may result in a difference between inflow and outflow for a given channel, especially during periods when variations of flows are experienced;
- losses to seepage can play a significant role for some streams; and,
- local runoff along the reach can add significant flow within a given time step.

It is obvious that a combined effect of the above three factors results in a nonlinear relationship between the flow in the channel at its tail node and at its head node. This functional dependence between the inflow and outflow into an arc representing a natural stream can be convex for some time intervals, and concave for others, depending on the hydrologic conditions in the basin. The effects of routing can be minimized by choosing longer simulation time steps. However, this reduces the accuracy of the analysis since the degree of natural flow variation may be drastically altered or completely lost in the process of averaging over weekly or monthly time steps. Mathematically, the flows at the upper and lower end of an arc (i,j) representing a natural stream can be expressed as x_{ij} and $f_s(x_{ij}, V_{ij})$ respectively:

$$y_{ij} = f_s\left(x_{ij}, V_{ij}\right) \tag{3.1}$$

where y_{ij} represent channel outflow at its downstream end and function f_s represents the effect of hydrologic channel routing for a given time step. Function f_s has two arguments – upstream inflow x_{ij} and the initial conditions in the channel represented symbolically by the initial volume V_{ij}. Linear programming allocation models have traditionally resorted to ignoring function f_s and assuming that $y_{ij} = x_{ij}$. This can only be justified by sufficiently extending the length of the simulated time step, which rules out the use models based on linear programming to assist in real time basin operation.

Non-linear constraints also exist on diversion canals. While the flow variation may not be as significant here, the maximum canal capacity is equal to the design flow rate provided that canal is in good operating condition. The lower bound is usually greater than zero for primary and secondary canals, since most of them require certain level of flow for successful operation of gates. However, there is often a non-linear relationship between the inflow into a diversion canal and the flow in the originating stream. This relationship defines the maximum flow that can be diverted. There are two types of diversions: controlled (gated) and uncontrolled. They impact the upper bound and the flow in a diversion canal, respectively. Both are examined in the following.

The relationship between flow Q and depth d in a river cross section is usually approximated using an exponential function of the form:

$$d = aQ^p \tag{3.2}$$

where d is depth, Q is flow, a and p are parameters determined by calibration. When elevation in the river is below the invert of the diverting weir, the diversion is not possible and the flow in the diversion canal equals zero. Hence, there is a minimum threshold flow in the river which must be available to operate a diversion canal. This is equally the case for gated and unregulated diversions. Once this elevation is above the invert, the diversion is possible but there is always the upper bound that can be diverted as a function of the flow in the river. This bound is dynamic. The gate operator can divert less or equal to the upper bound within each discrete time interval, depending on the way the gate is operated. Consequently, the model has to determine the upper bound on the diverted flow as a function of the overall solution for the whole system, since the flow in the stream supplying the weir is part of the overall solution. This can be expressed mathematically as:

$$u_d = f_d(x_s) \tag{3.3}$$

where u_d is the flow bound of the diversion canal and x_s is the available flow in the originating stream which supplies the weir. Similar constraints exist for ungated weirs. The difference is that inflow into this canal always equals the upper bound, so instead of the upper bounds, it is the value of the diverted flow that must be fixed as a function of the flow in the originating stream, hence

$$x_d = f_d(x_s) \tag{3.4}$$

where x_d is the diverted flow into the weir and the right hand side is the same as in (3.3). Note that both x_s and x_d are decision variables while f_d is a non-linear function which can usually be approximated with a polynomial.

The down side of using an iterative procedure with "successive linearization" of the above constraint functions are best demonstrated with an example. Consider a simple system depicted in Figure 3.3 with one reservoir, one irrigation block, one diversion channel and two river reaches. This example is based on a real-world constraint which exists on several weirs in the South Saskatchewan river basin in Southern Alberta. The system is solved with the WRMM model which relies on the Out-of-Kilter network LP solver and successive iterations. Solutions are derived over weekly time steps and iterations within one time step is presented in the following.

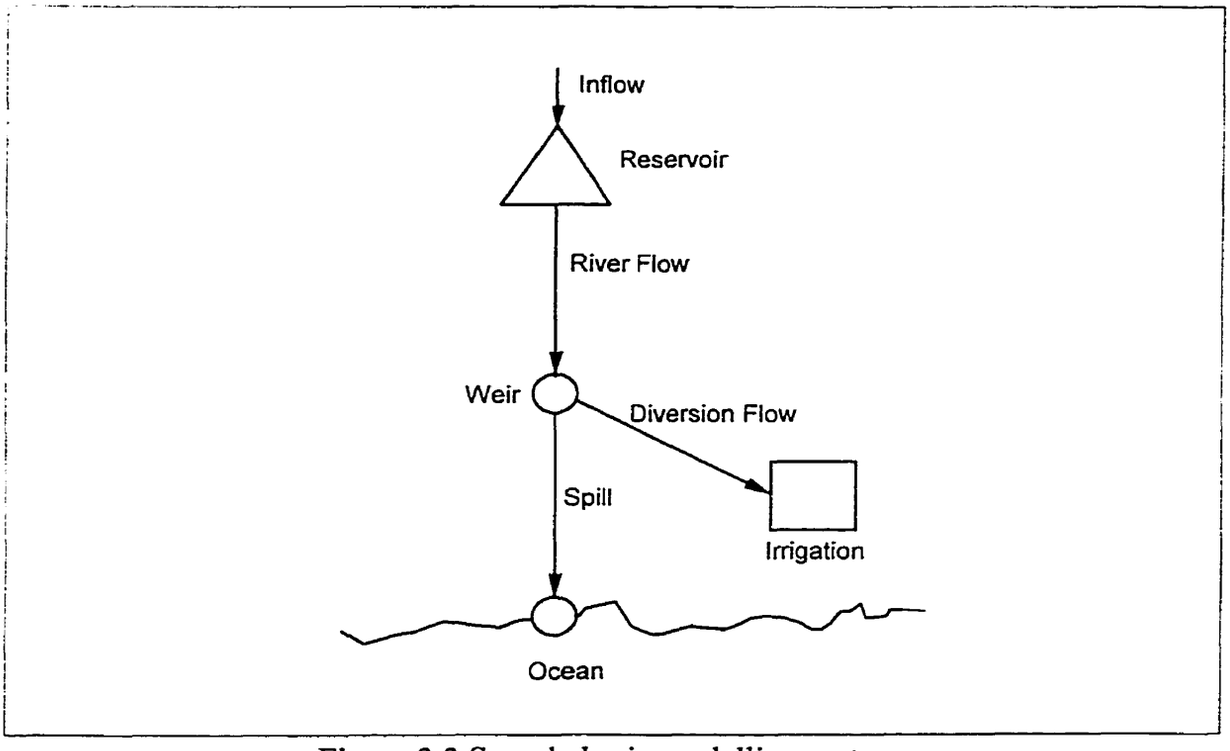


Figure 3.3 Sample basin modelling system

Note that there are no restrictions imposed on reservoir outflow, but there are restrictions on the diversions from the river at the weir. Maximum diverted flow is a function of the flow in the river according to the relationship depicted in Figure 3.4 below.

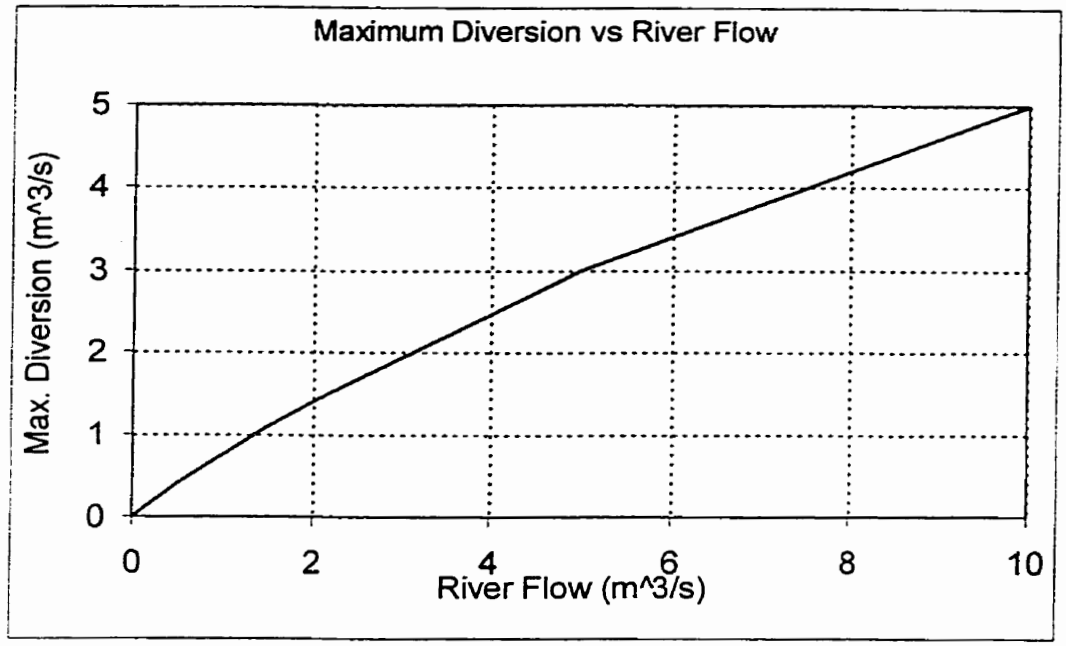


Figure 3.4 Maximum diversion vs river flow

As stated earlier, linear programming algorithms require that all upper bounds on flow be specified as fixed values for each simulated time interval. The above constraint is just one of many examples when this is not the case. To apply linear programming solvers with non-linear constraint functions that look like the one in Figure 3.4, it is necessary to go through the following iterative steps:

a) assume a set of fixed values for river flow and calculate the corresponding diversion flow limit;

b) submit the problem to the solver using the values of the upper bounds for diversion calculated in step a);

c) check if the solution flows derived in step b) comply with the function in Figure 3.4 within a given tolerance limit (e.g. +/- 1%).

d) If the check conducted in step c) failed, reset the upper bound of diversion flow based on the river flow obtained from the solver in step b) and repeat steps b) and c).

e) If the check conducted under c) was satisfactory, declare the problem solved and move on to the next time interval.

Assume that the reservoir is full, inflow is zero and the irrigation demand is 5 m^3/s. There is a small penalty (cost factor) for storage deficits, and a large penalty for irrigation deficits, which describes an allocation policy in this case (i.e. storage should always be released for irrigation). The best solution for a time step is easy to see: the reservoir release should be 10 m^3/s, with 5 m^3/s being diverted and 5 m^3/s being spilled into the ocean. The initial setting of the bounds is therefore 10 for the river channel below the reservoir, 5 for diversion and 10 for spills. However, since the linear programming objective function is to minimize the total system deficit (i.e. deficit in irrigation supply and deficit in storage), the LP derived solution for the first iteration is 5 m^3/s to river flow, 5 m^3/s to diversion canal and zero to spill. Solvers based on linear programming are unable to "see" that the flow in the river must be higher than 5 m^3/s to allow for a diversion of 5 m^3/s. All they see are the fixed bounds of 10 and 5 m^3/s for each arc in the network. Naturally, with this kind of problem representation spilling into the ocean seems unnecessary and detrimental to the objective of conserving storage. Yet this is exactly what is needed to arrive at the best possible solution. This will become apparent as the next few iterations are investigated. With the first solution derived by the model, the process proceeds to step c) where flow in the diversion canal is checked with the flow in the river. The model finds that the flow in the river derived in step b) was 5 m^3/s, and it finds that for this kind of flow the maximum possible diversion is 3 m^3/s. The problem is then sent back to step b) with the new limit on diversion flows set to 3 m^3/s. The new solution from the Out-of-Kilter algorithm then becomes 3 m^3/s to river flow, 3 m^3/s to diversion canal and zero to spill. Again, a check is made and the river flows of 3 m^3/s are found to correspond to 1.933 m^3/s. The process continues in this fashion from one iteration to another, as depicted in Table 3.1 until the final convergence is achieved only when the flows in the river channel and the diversion canal have both reached values close to zero. Hence the linear programming with "successive linearization" resulted in a solution with no storage release and 100% deficit for irrigation block while the storage was full and its cost factor was much lower than the cost of deficit at the irrigation block.

Table 3.1 Results of 14 successive LP solutions

Iteration	River Flow	Diversion Flow
0	10.000	5.000
1	5.000	5.000
2	3.000	3.000
3	1.933	1.933
4	1.360	1.360
5	1.002	1.002
6	0.751	0.751
7	0.577	0.577
8	0.456	0.456
9	0.366	0.366
10	0.294	0.294
11	0.237	0.237
12	0.190	0.190
13	0.153	0.153
14	0.123	0.123

The conclusion from the above example is the successive linearization may not work when flow constraints are non-linear, and in some cases it gives a solution which is far from optimal.

3.1.2 Reservoir Outflow Constraints

Reservoir outflows are limited by the capacity of the outlet structure (which can be either a weir or an orifice, but with similar effects). Reservoir elevation H_r determines the maximum possible outflow from the reservoir at each point during the given time period. Yet this elevation during the time period is also a decision variable resulting from the overall reservoir balance equation. For a given outlet and any reservoir elevation H_r, the maximum outflow capacity is an exponential function of H_r, i.e.

$$u_s = aH_r^p \tag{3.5}$$

where u_s is the upper bound on flow in the downstream channel while a and p are coefficients

which depend on the type of outlet. Note that on most reservoirs there is usually more than one outlet structure -- spillway for handling high flows and bottom outlet used in regular operation. At certain times both spillway and bottom outlet may be operated simultaneously. Also, in addition to the natural outflow, there may be one or more diversion canals supplied by the same reservoir with their unique outlet structures represented by their own functions similar to expression (3.5). Additional difficulty is that expression (3.5) shows the functional dependence for only one point in time when reservoir elevation equals H_r. Within a calculation time step (which can range in the order of days) H_r is not constant, and its value at the end of the calculation time step is part of the overall flow solution for the whole network. In the first approximation, one could infer that the average outflow capacity over a time step is the integrated average from the beginning to the end of the time calculation time step. If H_r is expressed as the flow in reservoir arc x_r for a given time interval, then

$$u_s = \frac{1}{T_e - T_i} \int_{Ti}^{Te} f_r(x_r) dt \tag{3.6}$$

where T_i and T_e are the starting and ending times for a given time interval and x_r is the sum of flow in reservoir arcs while $f_r(x_r)$ represents the function on the right hand side of expression (3.5). Only $f_r(x_r)$ at time T_i is known. The upper bound u_s can either be increased or decreased within a time interval, depending on the flow solution which may include reservoir refill or release. Expression (3.6) determines the upper bound on the outflow. Depending on the downstream demands and the available runoff, the actual release may be less than the upper bound determined by (3.6) when the outlet structure is controlled by adjusting the gate openings. When there is no gate associated with the outlet structure, the above expression not only determines the maximum flow, but rather determines the actual outflow x_s:

$$x_s = \frac{1}{T_e - T_i} \int_{Ti}^{Te} f_r(x_r) dt \tag{3.7}$$

where x_r on the right hand side of equation (3.7) is the sum of flow in the reservoir arcs

which represents the storage change for a time interval. Inflow into the reservoir is part of the overall flow solution for a given time step, and it also impacts the storage change within a time step. Equation (3.7) is a non-linear constraint imposed on outflows from reservoirs which have ungated outlet structures.

Consider a numerical example depicted in Figure 3.5 showing a simple system with one reservoir and two outflows. One outflow provides supply to a municipality with a maximum flow governed by reservoir elevation according to the functional relationship in Table 3.2, while the other outflow is a large capacity bottom outlet for irrigation supply, capable of completely emptying the reservoir within a week.

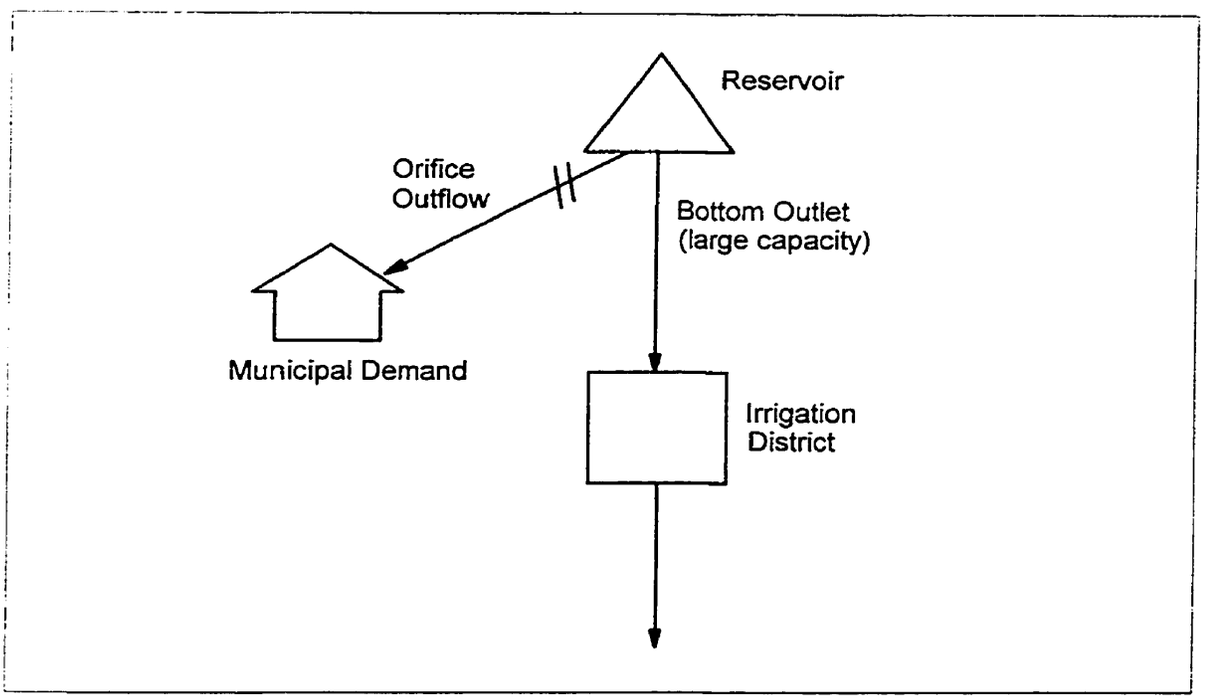


Figure 3.5 Example of outlet structure flow limitations

Assume that the bottom outlet flow limit is equal to the irrigation canal capacity of 50 m^3/s, while the orifice outflow limits are a function of reservoir elevation as shown in Table 3.2. The example presented in Figure 3.5 has the following cost factors: municipal demand is the most important, with a cost factor of 500 per 1 m^3/s of deficit flows, followed by the irrigation

block with a cost factor of 10 per $1m^3/s$ of deficit flows, and finally deficit in reservoir storage is assigned a cost factor of 1 per $1m^3/s$ of deficit in the units of flow. Storage is converted to flow by dividing volume with the length of the time step, which in this case is assumed to be 7 days.

Table 3.2 Technical description of the reservoir outflow test problem

VOLUME ($1000m^3$)	ELEVATION (m)	OUTFLOW (m^3/s)	ELEVATION (m)
0.	1653.540	0.000	1660.000
195.082	1654.230	2.350	1661.225
406.260	1654.920	3.678	1662.450
631.603	1655.609	4.954	1663.675
871.882	1656.299	6.029	1664.900
1125.811	1656.989	6.977	1666.125
1395.449	1657.679	7.834	1667.350
1679.896	1658.369	8.622	1668.575
1978.765	1659.058	9.355	1669.800
2290.896	1659.748	10.044	1671.025
2620.025	1660.438		
2961.258	1661.128		
3315.754	1661.818		
3682.611	1662.508		
4059.384	1663.197		
4448.518	1663.887		
4847.568	1664.577		
5258.979	1665.267		
5680.305	1666.000		
6850.000	1668.000		

RESERVOIR INFLOW 12 m^3/s
MUNICIPAL DEMAND 7.5 m^3/s
IRRIGATION DEMAND 20 m^3/s
STARTING RESERVOIR LEVEL 1667 M

PENALTIES PER 1m3/s FOR DEVIATING FROM IDEAL CONDITION:
MUNICIPAL DEMAND PENALTY = 500
IRRIGATION PENALTY = 10
STORAGE PENALTY = 1

The above is a demonstration of a mathematical program which has a linear objective function and non-linear constraints. The next few lines will examine the results of applying an LP solver to this problem in an iterative manner:

a) the initial outflow capacity for orifice outflow is set to the initial reservoir level of 1667 m, which corresponds to about 7.5 m^3/s. This capacity is equal to the municipal demand;

b) the solution derived by the model results in an empty reservoir since reservoir storage and inflow are less than the sum of both municipal and irrigation demand which get 7.5 m^3/s and 14.87 m^3/s, respectively;

c) the model then calculates the average orifice outflow capacity based on time-integration of reservoir levels for the entire week by starting from 1667 m and calculating new elevation (and the corresponding maximum outflow) at the end of each day assuming steady state inflow of 12 m^3/s and outflows listed under b); and,

d) the new outflow capacity of 2.85 m^3/s obtained in step c) is checked with the assumed outflow capacity of 7.5 m^3/s and since there is a large difference between the two the process is repeated from step a) assuming the outflow capacity of 2.85 m^3/s instead of the initially assumed 7.5 m^3/s.

The final solution renders municipal outflow of 2.85 m^3/s, irrigation supply of 19.51 m^3/s with an empty reservoir which corresponds to deficit in storage of 10.359 m^3/s. The objective function (total cost of deficit) of this solution is:

$$\text{Total cost} = (7.5 - 2.85) \times 500 + (20 - 19.51) \times 10 + 10.359 \times 1 = 2340.26 \qquad (3.8)$$

Deficits are calculated in brackets as the difference between the stated target and the achieved supply. It is easy to see that the above solution is far from the best. A much better solution can be picked up manually, by assuming that the reservoir stays full during this time interval. This would result in 7.5 m^3/s allocated to municipal demand and 4.5 m^3/s to irrigation, without any storage deficit. The corresponding value of the objective function is then:

$$\text{Total cost} = (7.5 - 7.5) \times 500 + (20 - 4.5) \times 10 + 0 \times 1 = 155.00 \qquad (3.9)$$

This is certainly a much better solution in terms of minimizing the objective function than the one obtained previously. The iterative use of LP fails to deliver a high quality solution. This is because LP only takes into account the fixed value of the channel upper bound, be it 7.5 in the first iteration or 2.85 in the second, along with the pricing vector, which implies that storage should yield to both municipal demand and irrigation. LP solvers cannot address a relationship between emptying storage for irrigation and the resulting change of the limit on flow capacity for municipal supply.

3.1.3 Hydropower Flow Constraints

The following discussion refers to hydropower plants which operate with variable head-flow combinations. For a given time interval, the net head H_n is determined by subtracting from the upstream reservoir elevation the tail water elevation and the appropriate head loss across the plant. The relationship between the power and flow is given by:

$$P_h = H_h . Q . \gamma . \eta_h \tag{3.10}$$

where P_h (MW) is the produced power, H_h (m) and Q (m^3/s) are the average net head and the average flow over a time interval, γ is the specific weight of water (9806 N/m^3) and η_h is the power plant efficiency factor. Hence, a specified demand for power over a given time interval cannot be converted to the units of flow without knowing the average net head, which is a function of the overall reservoir balance in the final solution, since it depends on the average headwater elevation as well as the average tailwater elevation for a time step. Average headwater elevation is an integrated average between the starting and the ending reservoir level for a time step, while the tailwater elevation may be either a non-linear function of flow for the downstream channel or the result of the reservoir balance of a downstream reservoir which defines the tail water elevations by its lake level.

Note also that Q, H_h and η_h are mutually dependent, which means that if one of them is fixed it determines a unique range of values for the other two. Assuming that the required power is known at the beginning of the time step, the flow in reservoir arcs determines the average

net head H_h over a time step, which in turn sets the flow requirement Q on the right hand side of equation (3.10) such that desired power P_h can be generated. Therefore, if the power target is known, it is possible to setup an iterative scheme and use LP, although the dangers of doing so are similar to those addressed in Sections 3.1.1 and 3.1.2.

More importantly, the hydro power target is often not known, and the hydro power plants may appear in the objective function in the following form:

$$\text{maximize} \sum\sum H_h.Q.\gamma.\eta_h \tag{3.11}$$

where the first summation is over all time intervals while the second summation is for all hydro power plants in the basin. The above term may appear as only one item in the objective functions, where other items may be related to other water management objectives, such as irrigation, industrial and municipal water use, riparian flow requirements, etc. With complex non-linear relationships between head and flow for various hydro power plants withing one time step, as well as across multiple time steps, it is difficult to resort to the use of linear programming. Previous efforts to apply successive linearization to this problem have been attempted (Sun et al., 1995) but they have gone on largely without comparing the results to global optimums, since a reliable non-linear solver with high likelihood of finding a global optimum for these problems has yet to be established.

3.1.4 Return Flows from Irrigated Blocks

Return flows associated with irrigation or other consumptive use are usually expressed as a percentage of gross diversion within a time interval, although a constant return flow factor may be added to the percentage. The value of actual percentage may vary during the season, but the variation may be considered known as it is based on empirical observations. There may be one or more return flow channels associated with a single irrigation block. Each channel returns a portion of the flow diverted into a block at one or more different points of return in the system. Irrigation return flows are represented by arcs with both upper and

lower bounds set to the same value equal to the fraction of the gross diversion, where gross diversion is part of the overall network flow solution. This can be expressed mathematically as:

$$u_r = l_r = f_g . x_d \tag{3.12}$$

where u_r and l_r are the upper and lower bound of a return flow arc, respectively, f_g is the (constant) fraction of gross diversion x_d that is returned to the system. Constraint (3.12) is linear, however it cannot be included in the minimum cost flow problem as formulated in with expressions (2.3), (2.4) and (2.5), especially since more than one return flow channel can be associated with the same irrigation block. Consequently, it is considered as a *non-network constraint* and as such it requires more general solution algorithms (McBride, 1985). The use of standard network solution algorithms such as the Out-of-Kilter algorithm still requires the use of iterations in addressing the return flow constraints.

3.2 A Review of Network Models in River Basin Management

A large number of various computer models have been developed since the early 1970s in an effort to aid river basin planning and management. A comprehensive review of these developments was compiled by Yeh (1985). The most widespread approach was based on using the network representation as depicted in Figure 3.1 and solving the corresponding linear minimum cost flow problem. The models which utilized these concepts are SIMYLD (Evanson and Mosley, 1970), ACRES (Sigvaldason, 1976), MODSIM3 (Labadie et al, 1986), WASP (Kucera and Dimnet, 1988); DWRSIM (Chung et al., 1989), CRAM (Brendecke et al., 1989), KCOM (Andrews et al, 1992) and WRMM (Ilich, 1993). Non-linearities associated with the bounds were handled by using longer computational time steps and by applying successive iterations within a time step if necessary. As stated earlier, this is done by initially guessing the bounds, solving the minimum cost flow problem, evaluating the network flow solution against the assumed bounds, re-setting the bounds to new values based on the previous solution, and re-iterating if necessary until the assumed bounds and

the network flow solution were within a reasonable tolerance limit. This process is repeated simultaneously with reservoir outflow, irrigation return flows, hydro power component and in some cases the channel time lag, which is also a function of the overall flow solution (Alberta Environment, 1995) which creates difficulties in the convergence process, requiring sophisticated convergence algorithms. It should be noted that each time an iteration is performed, a slightly different problem is submitted to the optimizer resulting in a new solution that becomes the starting point for the next iteration. There is no guarantee that this process will result in a convergence to the global optimum, as demonstrated by the previous numerical examples. The problem being solved is non-linear in terms of its bounds, and the guessing process solves successive linear approximations of a non-linear problem.

Kucera (1988) used a primal simplex method to address the multi-period planning procedure for a multi-reservoir system. Problems involving network and non-network linear constraints have been solved efficiently for long planning periods using the EMNET solver, which is claimed to be the fastest embedded generalized linear programming solver available (Sun et al., 1995). However, if iterations are employed within any of the above models their effect on the quality of the final solution has yet to be addressed. The non-linear constraints can be ignored by avoiding iterations, but with similar effects on the solution quality.

4 PROPERTIES OF FEASIBLE CIRCULATIONS

The problem of finding a feasible circulation in the network has two important aspects. Firstly, if it can be proven that no feasible solution exists for a given problem, the search for an optimal solution would cease. Secondly, since an optimal solution is necessarily feasible, it is instructive to gain insight into the properties of feasible solutions. This may help create algorithms which are capable of conducting a search exclusively through the feasible region. Some algorithms converge to an optimal point from both feasible and infeasible region of the search space. This may work well in certain cases, but in many instances this approach fails to find global optimum and lacks reasonable efficiency. This research will attempt to develop an approach to conduct search exclusively through the feasible region.

The theory of graphs defines feasible flow as a vector x_{ij} which satisfies constraints (2.3) and (2.4). The property of feasible flows (circulations) are examined first for unbounded circulations, then for problems with linear bounds where l_{ij} and u_{ij} are constant for all arcs (i,j). The discussion is then extended to problems with non-linear bounds. Note that non-linearity is here concerned only with the bounds. The shape of the objective functions has no impact to the issue of feasibility.

4.1 Unbounded Networks

In general, for unbounded networks ($l_{ij} = 0$ and $u_{ij} = +\infty$) the circulation vector is subject to general vector operations such as addition and scalar multiplication. For example, if a rational number is denoted by g and a feasible circulation vector by x_{ij}, then:

$$g(x_{ij}) = (gx)_{ij} \tag{4.1}$$

Hence a new feasible circulation vector can be generated by scalar multiplication of each element of x_{ij}. Similarly, if x_{ij} and y_{ij} are two feasible circulations while g and λ are two rational numbers, then:

$$g(x_{ij}) + \lambda(y)_{ij} \tag{4.2}$$

is also a feasible circulation. It is easy to see that the transformations do not violate conditions (2.3) and (2.4) for unbounded problems when ($l_{ij} = 0$ and $u_{ij} = +\infty$). The above vector operations allow generation of new feasible solutions as a linear combination of two or more existing feasible solutions. Similar operators can be applied for bounded problems provided that maximum and minimum flows in the network are known, as discussed in the following.

The other important property of circulations, known as the *flow decomposition principle*, states that any circulation can be decomposed into a finite set of directed cycle flows. Let w be a set of all directed cycles in the network, and let the decision variable be flow along cycles $f(w)$. By introducing a mapping function α_{ij} which is equal to 1 if arc (i,j) is contained in a given cycle and 0 otherwise, the flow in arc (i,j) can be expressed as a sum of all cycle flows w' which contain arc (i,j). Then

$$x_{ij} = \sum_{w' \in w} \alpha_{ij} f(w') \tag{4.3}$$

Flow decomposition principle is based on the *fundamental cycle theorem* which states that *each set of directed cycle flows has a unique representation as arc flows* x_{ij}. *Conversely, every circulation* x_{ij} *can be represented (although not necessarily uniquely) as a set of directed cycle flows at most m directed cycles, where m is the total number of arcs in the network.* A deductive proof of this principle is as follows: Starting from a given circulation x_{ij}, find an arbitrary directed cycle and reduce its flow such that at least one arc flow on the cycle becomes zero. After repeating this m times, each arc flow has been reduced to zero and circulation has been decomposed into m directed cycle flows, represented by a flow reduction on each of the m cycles found in the process. The *Augmenting Cycle Theorem*, one of the most important theorems of network flows, extends this observation to the cycles which are not necessarily directed (Ahuja et al, 1993). A cycle w (not necessarily directed) in network G is called an *augmenting cycle* with respect to flow x_{ij} if by augmenting a

positive amount of flow *f(w)* along the cycle the flow x_{ij} remains feasible. The augmentation increases the flow on forward arcs in the cycle and decreases the flow on reverse arcs, so *w* is an augmented cycle in G if $x_{ij} < u_{ij}$ for every forward arc (i,j) and $x_{ij} > l_{ij}$ for every reverse arc. The mapping function α_{ij} can also be defined for cycles that are not necessarily directed, by assigning values of 1, -1 to the arcs which belong to the cycle with forward and reverse orientation, respectively, or assigning zero for arcs which do not belong to cycle *w*. Similarly, the usual way of handling lower bounds is the representation of the entire network is by the use of *residual* network. In residual networks each arc (i,j) is replaced by two parallel arcs with opposite orientation. The arc with the same orientation as the arc in the original network has the upper bound equal to $u_{ij} - x_{ij}$, while the arc with opposite direction has the upper bound set to $x_{ij} - l_{ij}$. Both arcs have lower bounds set to zero. This representation is consistent with representing variable *x* as a sum of two other variables *x'* and *x"*, which are easier to handle in terms of bounds since lower bounds are zero. With this representation, each augmenting cycle *w* in the original network G with respect to flow x_{ij} corresponds to a directed cycle in residual network G', and vice versa. Hence, fundamental cycle theorem can be applied on residual networks to account for lower bounds on arc flows. This opens a possibility of generating any feasible flow in the network by assigning flows along the augmenting cycles. Most of the existing network algorithms utilize this theorem in some form. Several algorithms have been developed with the sole purpose to identify the fundamental cycles in the network (Dorris and Chen, 1981). Once identified, the fundamental cycles can become building blocks for generating various feasible flows through the network.

4.2 Circulations with Linear Upper and Lower Bounds

While circulations in unbounded networks are guaranteed one simple starting feasible solution (zero flows on all arcs), that is not the case for networks with positive lower arc bounds. It is imperative to first establish that a feasible circulation exists for a given network prior to conducting a search for an optimal circulation. Infeasible solutions imply that

circulation vector which satisfies both (2.3) and (2.4) cannot be found. Expression (2.4) can be re-written as:

$$\sum_{\{j:(i,j)\in A\}} x_{ij} = \sum_{\{j:(j,i)\in A\}} x_{ji} \qquad \forall i \in N \tag{4.4}$$

Arc flows on the left side of the above equation represent all incoming flows into node i while arc flows on the right hand side represent all outgoing flows from node i. Add condition (2.5) to both sides of equation (4.4) to give

$$\sum_{\{j:(i,j)\in A\}} l_{ij} \le \sum_{\{j:(i,j)\in A\}} x_{ij} = \sum_{\{j:(j,i)\in A\}} x_{ji} \le \sum_{\{j:(j,i)\in A\}} u_{ji} \qquad \forall i \in N \tag{4.5}$$

Expression (4.5) is a condition for feasible flows through node i, and it states that the sum of the lower bounds of incoming arcs into node i must be less or equal to the sum of outgoing arcs for the same node. If this condition does not hold for any node in the network, there can be no feasible flow. Note that any connected group of nodes labelled as a set of nodes S in a given network forms a subgraph which can be viewed as a single node in relation to the rest of the network. Let S be such a set (also called a subgraph on N) and also define a subgraph S', such that S' = N \ S where N is the set of all nodes in the network. Arcs incident to nodes (S,S') and (S',S) are said to form a cut. The above expression can be written for subgraph S instead of node i as:

$$\sum_{\{(i,j)\in(S,S')\}} l_{ij} \le \sum_{\{(i,j)\in(S,S')\}} x_{ij} = \sum_{\{(j,i)\in(S',S)\}} x_{ji} \le \sum_{\{(j,i)\in(S',S)\}} u_{ji} \quad \forall(S,S') \tag{4.6}$$

The existence of a feasible circulation in a network requires that this relationship holds for all possible cuts (S,S') in the network. Conversely, the existence of one or more cuts (S,S') for which

$$\sum_{\{(i,j)\in(S,S')\}} l_{ij} \ge \sum_{\{(j,i)\in(S',S)\}} u_{ji} \qquad \forall(S,S') \tag{4.7}$$

is sufficient condition for infeasibility. This implies that all cuts (S,S') have to be examined using expression (4.7) to determine if the network has a feasible solution or not. This task

is included implicitly in most network algorithms associated with either finding the maximum flow through the network or finding the minimum cost flow. The net flow v across a cut (S',S) is a sum of all incoming and outgoing flows for (S',S). This is expressed mathematically as:

$$v = \sum_{\{(i,j)\in(S,S')\}} x_{ij} - \sum_{\{(i,j)\in(S',S)\}} x_{ij} \tag{4.8}$$

Since each of the terms on the right hand side of (4.8) is bounded, it can be replaced with arc bounds provided that the equation changes into inequality

$$v \le \sum_{\{(i,j)\in(S,S')\}} u_{ij} - \sum_{\{(i,j)\in(S',S)\}} l_{ij} \tag{4.9}$$

Expression (4.8) is also known as the *Maximum-Flow Minimum-Cut Theorem* (Ahuja et al., 1993). One important conclusion of this theorem is that every cut (S,S') in the network has a maximum and minimum net flow v associated with it which can be found by inspecting all of its cuts (S,S').

Consider now a circulatory network with a cut (S,S') which contains only the arcs incident to the system balance node. In other words subset S' contains only the system balance node while subset S contains all other nodes in the network. For this case the upper and lower bounds on v are defined by two specific (maximum and minimum) circulations that can be realized in a given network. Denoting those two circulations by Xmin and Xmax, it is obvious that:

$$\sum_{(i,j)} l_{ij} \le \sum_{(i,j)} X\min_{ij} \le \sum_{(i,j)} x_{ij} \le \sum_{(i,j)} X\max_{ij} \le \sum_{(j,i)} u_{ji} \quad \forall (i,j) \in A \tag{4.10}$$

Therefore, due to flow constraints associated with cuts, the actual flows on some arcs may never reach their upper or lower bound. The most they could ever reach is determined by the value of Xmin and Xmax on every arc. Finding the two circulations (Xmin and Xmax) is of paramount importance, since they explicitly define the limits on feasible flows in each arc.

Various algorithms are available for finding feasible flows Xmin and Xmax, which can be viewed as a set of two maximum flow problems for a network with the same incidence mapping but different arc bound structure, or it can be viewed as set of two minimum cost flow problems in the same network which differ only in terms of the sign of the arc costs. Whichever approach is used, feasible circulations Xmin and Xmax can be easily obtained using the existing network flow algorithms which are capable of starting with zero flows for all arcs, and are also capable of identifying infeasible solutions.

Assuming that Xmin and Xmax are available, the generation of other feasible solutions in a bounded network can be conducted in several ways. One way is to use linear combinations of Xmin and Xmax and the transformations of the circulation vector stated in (4.2). For example, a new feasible circulation X can be determined as

$$x = g(x_{\max}) + \lambda(x_{\min}) \tag{4.11}$$

provided that the following conditions are attached to f and λ:

$$g + \lambda = 1 \quad \text{and} \quad 0 \le g \le 1, \quad 0 \le \lambda \le 1 \tag{4.12}$$

Moreover, any circulation X obtained using expression (4.11) can be used as a basis for generating additional feasible solutions instead of the initial Xmax and Xmin. Another way of generating a feasible flow X is based on using the flow decomposition theorem (4.3), which allows subtraction of a directed cycle flow from maximum circulation Xmax or addition to minimum circulation Xmin. Let X(c) and X(p) be two directed cycle flows. Then

$$x = g(x_{\max} - x(c)) + \lambda(x_{\min} + x(p)) \tag{4.13}$$

is also a feasible flow in the same network, provided that the cycle flows X(c) and X(p) are within the limits related to arc bounds. For example, if arcs 2, 3 and 7 belong to cycle c, then the largest possible cycle flow $X_{max}(c)$ on cycle c is determined by:

$$X_{max}(c) = \min\{(X_{max}(2) - X_{min}(2)), (X_{max}(3) - X_{min}(3)), (X_{max}(7) - X_{min}(7))\} \qquad (4.14)$$

Cycle flow X(c) can have any value between zero and the maximum X(c) defined by (4.14).

4.3 Tucker's Representation of the Circulation Space

The basis of Network Simplex method is the relationship between maximum spanning trees (also known as *maximum forests*) and the incidence matrix. Denote with m the number of arcs in a network and with n number of nodes. Equality constraints expressed by condition (2.3) represent a set of continuity equations written for every node in the network. It is obvious that for a connected network, $m \geq n - 1$ since each node must be connected to another node via at least one arc. Typically, the number of arcs is greater than the number of nodes, particularly in the case in circulatory networks. This leads to an observation that equality constraints (2.3) are represented by a system of equations with an $m \times n$ matrix which has $m-n+1$ columns that are linearly dependent and can be expressed as a combination of other columns, while $n-1$ columns are linearly independent. It is known from graph theory that the search for linearly independent columns in an incidence matrix is equivalent to the search for a maximum forest in a network (Ahuja et al, 1993). There is a finite number of maximum forests in any network and each of them is a basis for the minimum cost flow problem in the Network Simplex algorithm, i.e. each one defines a maximum spanning tree. The pivoting operation in the Network Simplex method is equivalent to moving from one maximum spanning tree to another. By denoting the incidence matrix with **E** and a circulation vector with **X**, circulation can be expressed as

$$\mathbf{EX} = \mathbf{0} \qquad (4.15)$$

where **E** is an $m \times n$ matrix while **X** is a column matrix of size m. This system of equations can be solved for some of the variables X in terms of the others, and written equivalently, for various subsets of arcs $F \subset A$ as:

$$\mathbf{E}_F \mathbf{X}_F = -\mathbf{X}_{A \setminus F} \mathbf{E}_{A \setminus F} \tag{4.16}$$

Arcs which belong to subset F form a maximum forest while all other arcs excluded from the maximum forest are represented by A \ F where A is the set of all arcs in the network. Equation (4.16) is known as the *Tucker's representation of circulation space* (Rockafellar, 1984). The size of matrix $\mathbf{E}_F$ on the left side of equation (4.16) is *n-1*, and the size of matrix $\mathbf{E}_{A \setminus F}$ on the right side of equation (4.16) is *m-n+1*. For any arbitrary choice of arc flows on the right hand side of equation (4.16), the arc flows on the left side can be recalculated since it is known from graph theory that the left hand side matrix in (4.16) can be transformed into a triangular matrix and solved using direct substitution. Since the choice of arc flows on the right hand side of (4.16) is arbitrary, it can include integer values for some and mutual non-linear relationships for others. This gives way to generation of feasible solutions which would include mixed integer non-linear feasible solutions for *m+n-1* decision variables. In other words, if there is a non linear dependence between two arc flows on the right hand side of equation (4.16), for any selection of the value of independent variable the dependent variable could be recalculated.

So far the above analyses excluded constraints on arc bounds, however constraints are easy to include provided that Xmin and Xmax circulations are known. If the appropriate Xmin values are input on the right hand side, the recalculation of the left side will yield the remaining Xmin values thus completing the entire minimum flow vector. The case is the same if Xmax values are input on the right side of equation (4.16). Consequently, any set of arbitrarily chosen arc flows between Xmin and Xmax for any given arc would yield a feasible solution by recalculation of the remaining elements of the circulation vector on the right hand side of equation (4.16). A conclusion that should be emphasized at this point is that any feasible circulation X can be created using the knowledge from the network flow theory.

The above conclusions can be used as the basis for building new search algorithms for non-

linear network optimization that utilize various contemporary optimization strategies. When arcs bounds have finite values, the total value of feasible circulation is also bounded by maximum and minimum values.

The operator stated in expression (4.11) can only be used to generate a new feasible solution for a problem with linear flow bounds and non-linear objective function. Several researchers have taken advantage of this operator on problems with linear bounds and non-linear objective functions (Grafenstette, 1987; Michalewicz 1994) .

For non-linear flow bounds, flows Xmin and Xmax become functions of the individual arc flow values, hence it is necessary to keep track of them and update them in each step of the search process. This results in combination of network flow theory with the new search strategies. No previous research publications could be found that proposed the same concept. This approach will be explained in more detail in the following sections.

5 EVOLUTION PROGRAMS

5.1 Introduction and Literature Review

Evolution programs are probabilistic optimization algorithms based on similarities with biological evolutionary process. In this concept, a *population* of individuals, each representing a search point in the space of feasible solutions, is exposed to a collective learning process which proceeds from generation to generation. The population is arbitrarily initialized and subjected to the process of *selection*, *recombination* and *mutation* such that the new populations created in subsequent generations evolve towards more favourable regions of the search space. This is achieved by the combined use of the *fitness* evaluation of each individual and the selection process which favours individuals with higher fitness values, thus making the entire process resemble the Darwinian rule known as the *survival of the fittest*.

Terminology, notation and opinions about the importance and the nature of the three underlying processes (selection, recombination and mutation) vary throughout the research community. Back and Schwefel (1993) identified three main streams of evolutionary algorithms that have emerged in the last three decades: evolution strategies (ES) developed by Rechenberg (1965) and refined by Schwefel (1981); evolutionary programming (EP) developed by Fogel Owens and Walsh (1966) and recently refined by Fogel (1991), and *Genetic Algorithms* developed by Holland (1975) and refined by De Jong (1975), Grefenstette (1987) and Goldberg (1989). The field of evolutionary computation has evolved since the pioneering work of these researchers. Nowadays there are several well established international annual conferences on this topic attracting hundreds of participants while the number of papers describing specific applications is growing at an exponential rate. In spite of the lack of strong theoretical background, the evolutionary approach has emerged in the last two decades as a powerful and promising technique that has generated much interest in the scientific and engineering community, mainly as a result of numerous successful

applications which far surpassed other search methods in terms of their ability to deliver superior solutions. It is obvious that many different evolution programs can be formulated to solve the same problem. They could differ in terms of their data structure used to represent a single individual, recombination operators used for generating new individuals, the selection process, methods of creating the initial population, methods for handling the constraints of the problem, and the search parameters such as population size. Regardless of these differences, they all share the same principle: a population of individuals is subjected to selection and reproduction which is carried out from generation to generation until no further improvement of the fitness function can be achieved.

There are two large classes of problem representation, known as *binary* or *floating point.* Genetic algorithm propounded by Holland (1975) uses a fixed length binary string and only two basic genetic operators: cross-over and mutation. The raw power of genetic algorithms is demonstrated on a specific application related to selecting the best combination of 40 binary variables, which can be viewed as finding the best combination of 40 off and on switches in a control related problem. This outline provides insight into some of the difficulties related to the binary representation and reveals the need for a more suitable representation, which is addressed in the following.

5.2 Explanation of Genetic Algorithm using a Binary Problem

Most GA application to date have been applied on some form of a binary problem (Goldberg, 1987). This is acceptable if a decision variable represents a real world phenomena which has only two defined states (on or off). The term *genetic* comes from the basic idea to represent a possible solution of the optimization problem as a long binary string where each binary value is either 0 or 1, thus forming an *artificial chromosome* for one possible solution. The initial population of artificial chromosomes is random. Having created the initial population, the algorithm then proceeds by comparing all members of the initial population and ranking them from best to worst in terms of their fitness. A fraction of the best solutions

are retained and used to breed among themselves producing new generation of possible solutions using the *cross over* and *mutation* techniques which were initially designed to resemble similar processes in nature. The way artificial chromosomes are combined is very much the same as the way biological chromosomes are combined when offspring of any life form is created. The process thus continues from generation to generation and the natural selection of artificial chromosomes eventually results in convergence to an optimum. In other words, in technical optimization based on an evolutionary processes the survival of the 'parent' chromosomes depends on its fitness with respect to the objective function. This bias in favour of the solutions with better fitness generates offspring with a high likelihood that some of the individuals will surpass the fitness of their 'parents'. The process stops when there is no improvement in the value of the objective function of future generations compared to their parents.

The above approach is very efficient when the decision variable can take one of the two possible states. Their power and efficiency is reduced when they are applied to problems with a floating point decision variable with difficult constraints. Several attempts have been made to convert problems with floating point decision variables into equivalent binary problems, although with varying success (Koza, 1993).

Denote with a(i) a randomly generated binary string of length 40 with values of 0 or 1 in each string bit. For example, a(1) could look like this:

a(1) = 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 1 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 0

In this representation the values 0 or 1 describe the switch status (on = 1 and off = 0). Since the total number of switches in the system is 40, the number of combinations to be examined is 2^{40}, where number 2 represents the two possible states (on/off) for each switch. For each randomly generated solution represented by a binary string a(i), an objective function (total cost of pumping) can be calculated. The classical genetic algorithm proceeds as follows:

1. Generate randomly a population of 100 binary strings a(i);
2. Calculate the objective function for each solution generated in step 1 and rank the solutions from the best to the worst in terms of their optimality;
3. Select a small percentage (typically 5% to 30%) of the best solutions obtained in step 2 for further reproduction and discard the rest as 'unfit' for having offspring.
4. With the best solutions selected in step 3 as 'parents', generate a new set of 100 binary strings of 'offspring' using the cross-over and mutation operators.
5. Repeat the steps 2 through 4 until the calculated objective function value shows no further improvement in terms of optimality or when the improvement is within a specified small tolerance limit.

Many types of cross over and mutation operators have been tried by various researchers. In natural systems, a new organism is created by a random split in the chromosome string at numerous locations and mutual replacement of the genetic code in the resulting offspring. A simplified example of this is a single split point of a chromosome into two parts and mutual replacement. For example, consider cross-over strings b(1) and b(2) obtained as children of strings a(1) and string a(2) listed above with a split point at cell 11 (for simplicity of this example string a(1) has bits equal to 0 for the first 11 cells and equal to 1 for the remaining 29 cells):

a(1) = 1 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 0
a(2) = 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

b(1) = 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 0
b(2) = 1 0 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Note that string b(1) has the first 11 cells from string a(2) and the remaining 29 cells from string a(1), while string b(2) has the first 11 cells from string a(1) and the remaining 29 cells from string a(2). To avoid 'degenerate' offspring a mutation factor is also introduced, which

amounts to a small random change of some cells from 0 to 1 or vice versa. Goldberg applied the mutation factor to a small fraction of the population (typically 2% to 3%).

A few issues are apparent from this brief description: population size, the death/survival ratio, cross-over and mutation operators are all arbitrary and entirely dependent on the experience and judgment of the person conducting the study. It would therefore seem strange that with so many degrees of freedom and presumably required calibration this approach can be so successful. This is especially of interest in view of the fact that each of the bit strings generated in the way described above is not necessarily guaranteed to correspond to a feasible solution, due to the nature of the processes being control with the control switches. In fact, in many real world problems with the above representation most of the solutions generated in the above manner could be infeasible. Researchers had to resort to the use of a penalty function associated with infeasibility which is added to the objective function and which forces infeasible solutions to 'die faster' in the process of evolution. Goldberg (1987) demonstrates the immense power of genetic algorithms by showing that they manage to converge to an optimal solution after having gone through only 25 generations, each with a population of 100, hence only 2500 possible states were examined out of the search space of 2^{40}. However, when this problem is considered in the context of floating point variables, the decision variable can no longer be binary (0 or 1) but must instead take the form of a real number, i.e. $Xmin(i) \leq X(i) \leq Xmax(i)$, where X(i) is assumed to have the required accuracy of $0.000 \leq X(i) \leq 999.999$, such that each decimal digit would have to be converted to a binary format, so each decimal digit would require four binary cells. Hence, a possible solution for one switch would now have a binary string with a length of 36. Since there are 40 switches, the total length of binary string is 36x40=1440, and the total number of combinations is changed from 2^{40} to 2^{1440} which clearly puts the problem in a different perspective.. The need to abandon the binary string representation of the problem has been recognized in the scientific community for some time (De Jong, 1988). Michalewicz (1994) advocates the use of "proper (possibly complex) data structures for chromosome representation together with an expanded set of genetic operators". He speculates that De Jong's historic work on the

theoretical formulation of the schemata theorem has inspired subsequent researchers to take his work like a gospel in spite of overwhelming evidence that binary representation was awkward in many applications. Instead, it is argued that 'natural' representation of a potential solution for a given problem and that it is a promising direction for further research. Koza (1993) observed that "representation is the key issue for genetic algorithms", and that their failure in many applications deals with the inability of the binary domain representation to deal with nontrivial constraints. Many other researchers agreed, hence the widespread propagation of titles such as *A Modified Genetic Algorithm* (Michalewicz et al., 1992), *Specialized Genetic Algorithm* (Janikow and Michalewicz, 1990) or a *Non-Standard Genetic Algorithm* (Michalewicz et al, 1991). Glover (1987) and De Jong (1990) were also critical of binary representation and suggest search for better domain representation. There is a widespread emerging belief that problem specific knowledge must be incorporated in the algorithm to ensure its efficient operation (Grefenstette, 1987; Davis, 1991; Forrest, 1993). The current state of the art in the field of Evolution Programs can thus be described as follows:

- There is no general algorithm applicable to all problems;
- Their efficiency varies from very efficient to inefficient as a function of problem size and complexity;
- Most evolution programs converge to an optimal point both from inside and outside of the feasible region, which means that often times more than 99% of the search effort is wasted on generating solutions that are infeasible;
- Evolution programs do not take into account shape or gradient of the objective function and they can conduct search within entire feasible domain, which gives them a better chance to find a global optimum;
- Evolution programs usually require calibration of the search parameters to ensure efficient convergence.

5.3 Numerical Example with a Floating Point Variable

This research is based on a variant of Genetic Algorithm with the following properties:

a) floating point domain representation, which means that chromosomes are represented with decimal numbers;

b) massive initialization procedure which uses a Monte Carlo random search to find the small initial parent population of high quality;

c) multi-parent crossover as proposed by the Genitor algorithm (Whitney, 1989); and,

d) properties of feasible flows in networks are included in the algorithm such that the search is always restricted to the feasible region by obeying the capacity and flow continuity constraints.

Item d) is explained in more details in Chapters 6 and 7. Items a), b) and c) are demonstrated in the numerical example below. Consider for example the problem of finding the best fit analytical equation of the outflow vs elevation curve given with ten pairs of (x,y) points in Table 3.2. A typical empirical equation for this curve is:

$$Q = AH^b \tag{5.1}$$

where Q is flow (m^3/s) while H is the net head (m) above the invert of the outlet, hence in the case of the curve given in Table 3.2 the net head is reservoir elevation minus 1660 m. Parameters A and B should be determined in such a way that the difference of the sum of squares between the analytic and tabulated values of flow for all 10 points is minimized. This can be formulated as: find the values of parameters A and b such that the value of the following objective function is minimized:

$$\min \sum_{i=1}^{10} (Q_i - A(H_i - 1660)^b)^2 \tag{5.2}$$

Values of Q_i and H_i are provided in Table 3.2 for each of the ten points. In addition, from other empirical studies related to similar curve fits it can be assumed that the most likely

range for the values of parameter b is (0,1) and for parameter A is (0,10). To be on the safe side in this example the values of parameter A are inspected in the range of (0,20). The value of parameter b must be less than 1 since it is never a straight line, and it must be greater than 0 since the values below zero would not result in an increasing function, while it is known that the outflow does increase with the increase in net head. Taking into account this simple knowledge about the problem reduces the search space to a value for parameter A in the interval (0,20) and the value of b in the interval (0,1) which has a significant impact on the solution efficiency. In some ways this can be compared to optimizing network flows with flow variables in each arc being restricted in value between a given minimum and maximum. The algorithm first generates randomly 5 possible solutions, as depicted in the top of Table 5.1. It then goes through a process of initialization, whereby additional 95 solutions have been generated in a pure random manner using the Monte Carlo approach and assigning random values to parameters A and b within the prescribed range for each. After each new solution is created, its objective functions is evaluated and compared to the worst objective function of the initial five solutions. If the new solution has better fitness than the worst of the five initial parents, the worst parent is discarded from the top five (also referred to as the "mating pool") and the new individual is included in it. At the end of the 100 trials, the mating pool has 5 members which all have a higher fitness values than the initial five randomly generated solutions at the start of the process. These parents will form the mating pool and they will be used as genetic material from which the new offspring will emerge.

The final and the most important part of the algorithm is recombination. A new solution is created by borrowing parameters A and b from one of the parents in the mating pool and applying a small modification using a normalized variate with a standard deviation equal to 1% of the value of the selected parameter. Denoting with a drand(1,5) a discrete function that selects an integer number between 1 and 5 with equal chance, and using A(i) and b(i) to denote the values of parameters A and b that are currently included in the mating pool, a new solution (values of parameters A and b) are generated using a variant of the standard GA procedure commonly known as recombination and mutation, shown in the pseud code below:

```
Procedure Recombination and Mutation
1     select = drand(1,5)
2     Anew = A(select) + N(0,A(select)*0.01)
3     if(Anew > 20) Anew = 20
4     if(Anew < 0) Anew = 0
5     select = drand(1,5)
6     bnew = b(select) + N(0,b(select)*0.01)
7     if(bnew > 20) bnew = 20
8     if(bnew < 0) bnew = 0
end
```

Line 1 assigns a random value between 1 and 5 to integer variable *select*. The new value of A is then assigned as being equal to one of the five existing values of parameter A that are already in the mating pool, with a small mutational change introduced by applying a normal variate with a mean of zero and standard deviation equal to 1% of the chosen value of A. Other small variates could have been chosen, the current choice is arbitrary and not critical to the search progress. Mutation is essential here since it generates the necessary diversity in the mating pool. Without mutation in this example the progress would soon be halted after the best of the combinations of A and b from the pool of five is found. However, it will be seen later that in some cases when the shape of the objective function is known, the mutation operator may not play such a big role.

Since the normal variate may result in variable Anew being above 20 or below zero, which is outside of the desirable bounds, it may be necessary to bring the value of Anew back within the desired bounds, which is done in lines 3 and 4. The entire process is repeated for parameter b, but with a different value of the variable select, which means that parameter A is picked from one parent in the mating pool while parameter b comes from another parent. As in the initialization procedure, each time a child is created with a fitness that is better than the fitness of the worst parent, the child is included in the mating pool and the worst parent is discarded from it. From there on the child continues to pass its genetic material to new generations. Table 5.1 shows the results of the objective function with parameters A and b for the best 5 solutions after 500 new individuals have been generated using recombination

and mutation process described above.

Table 5.1 Demonstration of a Genetic Algorithm

The first five randomly generated guesses:

Objective Function	A	b
443.816	0.025	0.564
1045.883	3.866	0.809
3885.054	11.700	0.480
9064.114	7.006	0.896
32137.926	16.457	0.747

The best five out of one hundred generated guesses:

Objective Function	A	b
6.294	3.111	0.504
29.131	2.199	0.743
32.011	5.054	0.144
36.606	2.275	0.455
48.131	5.518	0.273

The best five solutions after two hundred recombinations:

Objective Function	A	b
0.040	2.107	0.655
0.041	2.091	0.658
0.045	2.084	0.658
0.046	2.077	0.660
0.048	2.154	0.644

By comparison, this problem has also been solved with the Excel solver to give A=2.109, b=0.654 and the value of the objective function of 0.0395. To obtain this solution with the Excel solver, the settings must include higher precision requirements than those that are setup as default within the solver.

The new approach proposed in this research is the initialization procedure. Genetic algorithms typically start from any randomly generated set of parent solutions and converge eventually using recombination and mutation as described above. However, starting from good quality parents can be very beneficial. Consider for example the same problem as the one above, but solved without the initialization procedure, i.e. the first five solutions shown on top of Table 5.1 are used as parents that are subjected to recombination and mutation.

The results of this process are in Table 5.2, which shows the makeup of the mating pool after each 500 new individuals were created using recombination and mutation.

Table 5.2 Genetic Algorithm without initialization

The first five randomly generated guesses:

Objective Function	A	b
443.816	0.025	0.564
1045.883	3.866	0.809
3885.054	11.700	0.480
9064.114	7.006	0.896
32137.926	16.457	0.747

The best five solutions after 500 recombinations:

Objective Function	A	b
19.505	3.602	0.470
32.043	0.365	1.480
32.045	0.338	1.480
32.519	0.332	1.484
32.562	0.331	1.490

The best five solutions after 1000 recombinations:

Objective Function	A	b
7.508	1.048	0.994
7.665	1.008	0.998
7.736	1.004	0.999
7.804	1.024	1.003
7.854	1.009	0.985

The best five solutions after 1500 recombinations:

Objective Function	A	b
0.447	1.814	0.726
0.539	1.784	0.732
0.608	1.813	0.734
0.615	1.773	0.740
0.627	1.761	0.742

The best five solutions after 2000 recombinations:

Objective Function	A	b
0.050	2.065	0.665
0.063	2.036	0.671
0.065	2.038	0.669
0.067	2.030	0.672
0.067	2.030	0.671

The best five solutions after 2500 recombinations:

Objective Function	A	b
0.041	2.096	0.657
0.041	2.091	0.658
0.041	2.091	0.658
0.042	2.088	0.658
0.042	2.085	0.660

The proposed GA is stable, i.e. it does converge to the same solution eventually, however the effort is much more significant if the starting search points are of poor quality in terms of their fitness values. Compared to the 2500 generated solutions in this case, the initialization process resulted in the same final solution after only 300 generated trials.

To show that GA is not a pure random search, the reader should examine the impact of pure random search based on Monte Carlo generation of 10000 pairs of parameters A and b and recalculating the objective function (5.2) for each pair. The best 5 guesses and their values of the objective functions are listed in Table 5.3. The values of their objective functions are still three times larger than those obtained with GA.

Table 5.3 Top five guesses in Monte Carlo search afer 10000 trials

Objective Function	A	b
0.113	2.037	0.675
0.129	1.975	0.683
0.129	2.008	0.682
0.132	2.108	0.661
0.201	1.965	0.693

6 DESCRIPTION OF THE PROPOSED ALGORITHM

The proposed algorithm is intended to combine the properties of feasible circulations in order to develop evolution programs which conduct the search exclusively within the feasible region. This is done in an effort to ease convergence. Two approaches have been tested. The first was based on the use of fundamental cycle flows, and the second on the use of Tucker's definition of circulation space, which was found to be superior and it was therefore adopted in the three cases studies presented in Chapters 7, 8 and 9. The use of fundamental cycle flows was successfully tested on a pipeline optimization problem and published (Ilich and Simonovic, 1998). The algorithm outperformed a comparable gradient search method in terms of the ability to find global optimum. However, an evolution program based on the Tucker's definition of circulation space is presented in the following as a superior option.

Matrix equation (4.16) allows random selection of arc flows on the right hand side and recalculation of arc flows on the left hand side by substitution, since the left hand side contains the spanning tree structure and as such it can be transformed into a lower triangular matrix. One can pick any values for $m-n+1$ arc flows on the right hand side as long as they are within the respective bounds. Additional non-linear relationships related to the loss or gain of flow along the arcs which belong to the maximum spanning tree can also be included provided that the maximum spanning tree consists of arcs which all have the same orientation, or provided that the non-linear function F(x) which maps inflow into the arc into outflow has its inverse $F^{-1}(x)$. For many practical problems non-linear functional dependence can be approximated using a higher order polynomial so the latter is not too difficult to achieve using proper representation. The algorithm could then proceed with the following steps:

a) Determine existence of at least one feasible solution. If there is no feasible solution, stop and declare the problem infeasible. If there is a feasible solution go to step b).

b) Initialization. Generate randomly a large number of solutions (between 1000 and

5000, depending on the size of the problem) and select a small fraction (typically between 5 and 15) of the best solutions according to the value of their objective function. These will constitute "parent solutions" that will form the "mating pool" used in step c) to generate new solutions known as "offspring".

c) Recombination. Individual values of arc flows for each solution in the mating pool are considered as genetic material from which the new individual solutions are created. The process is continuous, and each offspring is compared to the worst ranked parent in the mating pool. If its fitness (value of the objective function) is better than the worst parent in the mating pool, the worst parent is discarded and the new offspring is placed in it. To discourage being trapped in local optimums, the algorithm must ensure that identical twins do not enter the mating pool, since they have a tendency to replicate themselves very quickly.

d) The process stops in one of the two ways -- when a specified number of individuals has been created or until the improvement of the objective becomes negligible within a given number of generated individuals.

Steps b) and c) above are critical for successful application of this algorithm. They are reviewed in more detail below.

6.1 Initialization

The massive initialization procedure in step 2 generates feasible solutions randomly, ensuring that the problem is addressed from all corners of the feasible region. Additional improvements in the initialization procedure are related to more frequent sampling of the points which are known to have a more favourable outcome. This is done in an effort to increase the likelihood of generating feasible solutions with good fitness values.

Consider for example a linarized cost function in Figure 6.1. This objective function is linear, so the flow in this component has a high likelihood of falling into one of the break points on

the curve in Figure 6.1, which are in effect the corners of the feasible region of interest to LP solvers. On the other hand, pure random sampling within the feasible range for this component (from 0 to x_{max}) has virtually no chance of ever hitting the exact value of any of the break points. One should therefore facilitate creation of suitable points in the search space by directing the random sampling to generate sufficient number of outcomes which coincide with one of the break points.

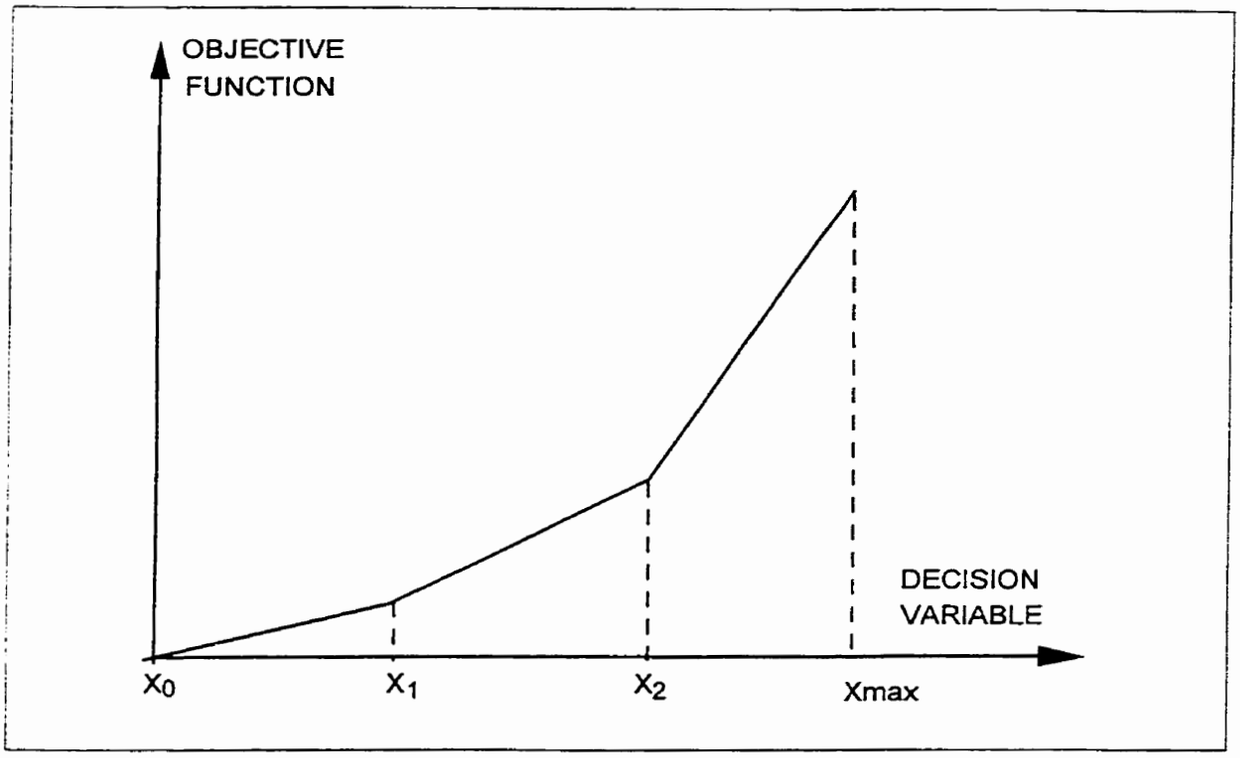


Figure 6.1 Sample objective function

The above objective function is related to a single arc flow. Assuming that function random() returns a pseudo random number between 0 and 1, the rules for generating random values for this arc flow in the initial population can be summarized as follows:

```
if(random() > 0.5)                  // 50% of generated solutions
      x = xmax random();            // have any value between 0 and xmax
else{       // 50% of generated solutions have one of x0, x1, x2 or xmax
separate = random();      // generated with equal likelihood
      if(separate < 0.25)
```

```
            x = x0;
        if(separate >= 0.25 && separate < 0.5)
            x = x1;
        if(separate >= 0.5 && separate < 0.75)
            x = x2;
        if(separate >= 0.75)
            x = xmax;
}
```

The above sampling approach ensures that there is sufficient number of points which belong to the corners of the feasible region in the initialization procedure. A similar approach can be used for other types of objective functions with known shape, such that random generation of solutions in the favourable region of the search space is encouraged. This is useful with functions that have multiple local minima for each variable, as demonstrated in Chapter 7, while Chapter 8 shows that the shape of the objective function is not the only parameter that can improve the quality of the initial population. Sometimes multiple regression and other functional relationships between two or more decision variables can greatly aid in the search for good initial parents, with reasonable expectations that good parents will generate better offspring faster, thus improving efficiency of the search process.

Once a maximum spanning tree has been identified, it allows for creating a sequence of solving balance equations for each node in the network. The process starts by selecting the nodes at the outer edges of the network, which have fixed inflows, and randomly assigning outflows for those outgoing arcs which are not part of the maximum spanning tree. Each time one outgoing arc has been assigned a value, the maximum flow bound for the next outgoing arc from the same node is updated accordingly by reducing the total inflow into a node with the outflow that has already been assigned, and comparing the balance of remaining inflow with the upper bound of the next outgoing arc. If the balance of the remaining inflow is smaller than the upper bound of the next outgoing arc, the upper bound is dynamically adjusted to equal the remaining inflow balance. Finally, when all outgoing

arcs which are not on the maximum spanning tree have been addressed in this manner, the remaining balance inflow is assigned to the outgoing arc on the spanning tree. This ensures both the preservation of flow continuity at each node, as well as compliance with the flow bounds. In water resources networks, typical decision variables are reservoir releases and diversions from the stream, while the maximum spanning tree is usually defined by the main river and its tributaries. Reservoir storage, diversion and return flow arcs form fundamental cycles and they are typically associated with independent decision variables.

Finally, sometimes it is necessary to consider a group of interconnected nodes acting as a single node, and the application of the above approach requires more knowledge of the nature of the network in order to adequately set up the search process. This is demonstrated in Chapter 7.

6.2 Recombination

Recombination and mutation are the main driving engines of genetic programming. To some extent the "gene therapy" mechanism explained below takes on a twofold role – as a mechanism that ensures that offspring is always feasible, and also as a mutation operator. The role of mutation seems to be dependent on the shape of the objective functions. Since a variety of shapes have been investigated in the non-linear transportation test problems, the role of mutation is addressed in more detail in Chapter 7.

Recombination is a process of combining genetic material (decision variables) from two parent solutions in an effort to create a new solution. In this algorithm the number of crossover points is the same as the number of independent decision variables. In fact, the entire genetic information describing a single individual is an array of all independent decision variables belonging to one parent solution. The mating pool is a database of a small number of parent solutions where each genome (complete solution) is stored as one database record. A complete solution may contain only independent variables, since the dependent

variables can be re-calculated. Consider an example in Table 6.1.

Table 6.1 Sample Five member Mating Pool with one possible offspring

Parent Solution a	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11
Parent Solution b	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11
Parent Solution c	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11
Parent Solution d	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
Parent Solution e	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11
Offspring Solution	x1=d1	x2=b2	x3=a3	x4=c4	x5=b5	x6=d6	x7=d7	x8=a8	x9=d9	x10	x11

Recombination operator proceeds as follows: starting from the most upstream node (e.g. node 1 in the above database), select randomly with equal likelihood one of the existing solutions for the first independent decision variable (arc flow) associated with this node. In other words, pick one of a1, b1, c1, d1 or e1 with equal likelihood. In the above table the selected variables (genes) for inclusion into offspring are shaded gray. For the first independent decision variable the selection was made from parent solution d and placed in the first column of the offspring solution.

Note that decision variable d1 has automatically affected the sum of inflows into its downstream node (x1=d1). Consequently, the inflows and the maximum outflows from that node will also be affected. Assume that the downstream node is node 2, and that an independent decision variable that was selected is from parent solution b as shown in Figure 6.1. The value of b2 (x2=b2) placed in the new offspring solution may now be outside of its flow bounds which were dynamically updated after node 1 was solved. If that is the case, i.e. if the value of randomly selected x2 is below the minimum or above the maximum defined by all other inflows and the corresponding minimum outflows from node 2, then the value of x2 needs to be adjusted such that it is brought within the bounds. The process of checking the compliance with the bounds and adjusting if necessary is termed "gene therapy". This adjustment will take place before the algorithm proceeds to set the value for x3. This approach has not been published prior to this research, and as such it constitutes

contribution to a wider community of investigators who deal with genetic algorithms. Gene therapy is re-visited in more detail in Chapter 7.

The progress in the search is based on placing each offspring that has better fitness value than the last ranked parent in the mating pool, and discarding this parent from the mating pool altogether. This provides a stable progression, since the quality of the initial mating pool can only be improved, it cannot deteriorate.

To summarize, both the initialization procedure and recombination are organized in such a way to ensure that only feasible solutions are created in the process. These ideas are explored in more detail in Case Study I in Chapter 7, which demonstrates the main features of the proposed algorithm. Numerical examples in Chapter 7 deal with a number of non-linear transportation problems that had previously been solved in the literature. Transportation problems can be viewed as a subset of network flow problems. Chapters 8 and 9 provide two applications of the proposed solution technique to water resources networks. They all vary in size and complexity, as outlined below:

Case Study I	problems of 49 and 100 variables, linear constraints, one linear and six non-linear objective functions;
Case Study II	416 variables (156 independent) and 988 non-linear constraints with linear objective function; and,
Case Study III	1628 variables (925 independent), 703 linear and 222 non-linear constraints with non-linear objective function.

7 CASE STUDY I -- AN EVOLUTION PROGRAM FOR NON-LINEAR TRANSPORTATION PROBLEM[1]

7.1 Introduction

As mentioned earlier in Chapter 6, in this model the first selection of the best individuals is conducted from the population created by a massive initialization procedure. The initial selection is later continuously updated to include a specified number of the best solutions found within a given number of trials. This approach is often described in EP terminology as $(\mu + \lambda)$ - ES (Schwefel, 1981) where μ parents produce λ offspring and the selection of the new parents is done from the best μ individuals selected out of all individuals generated in the process. The ranked-based replacement algorithm used in this research is similar to the one used in the Genitor algorithm (Whitley, 1989). This research has been inspired by previous efforts to apply EP approach to solving both linear and non-linear transportation problems (Vignaux and Michalweicz, 1989; Michalewicz et al., 1991). The same test problems that were solved by these researchers were used for investigating the proposed approach. This allows a comparison of solution quality and the required computational effort. Without the previous work, it would be hard to gauge the quality of the proposed approach given that traditional gradient based solvers often fail to provide good solutions to multi-dimensional non-linear problems.

A brief introduction to the balanced transportation problem is provided first, followed by a description of the evolution program broken down in five sections: initialization, evaluation, selection, recombination, and the gene therapy. The final section includes the definition of the test problems, discussion of the results for individual test functions, summary of comparison with the earlier results and references. Because of keeping the search within the

---

[1] Content of this Chapter has been accepted for publication in the Journal of Heuristics, Vol. 7(2), to appear in March 2001 issue (Ilich and Simonovic, 2001).

feasible region, the algorithm was termed Strongly Feasible Evolution Program (SFEP).

7.2 The Transportation Problem

Virtually every text book on operations research has some reference to transportation problems (TP), although most of them are associated with the available linear programming solution procedures (Hiller and Liberman, 1995). The problem is concerned with finding the least cost distribution policy for a shipment of a single commodity from m sources to n destinations subject to the capacities of each source s(i) and each destination d(i). This can mathematically be expressed as:

$$\text{minimize} \qquad \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}(x_{ij}) \tag{7.1}$$

Subject to:

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= s(i), \quad \text{for } i = 1, 2, \ldots m \\ \sum_{i=1}^{m} x_{ij} &= d(j), \quad \text{for } j = 1, 2, \ldots n \end{aligned} \tag{7.2}$$

The above defines the *balanced* transportation problem due to the equality sign in the constraints. Note that out of the total of $m\ n$ variables x_{ij}, $(m+n-1)$ are dependent and $(m\ n + 1 - m - n)$ are independent. When all objective functions $f_{ij}(x_{ij})$ are linear, the entire problem is linear. The linearity also assumes that all $(m\ n + 1 - m - n)$ independent decision variables x_{ij} are also independent of each other, i.e the value of one independent decision variable has no impact on the value of another. There is a large body of literature covering the well established solution procedures for the linear case. A general solution procedure for a non-linear case is still lacking.

7.3 Description of the Proposed Evolution Program

A floating point representation is used to describe all individuals (feasible solutions) which evolve in the solution procedure. A detailed description of the algorithm is divided in five sections: initialization, evaluation, selection, recombination and gene therapy. Of those, initialization, recombination and gene therapy are of particular interest.

7.4 Initialization

The process of initialization takes advantage of the fact that there are *(m+n-1)* dependent and *(m n + 1 - m - n)* independent variables. This means that one could assign values to independent variables in a random manner, and then recalculate the dependent variables to ensure compliance with the constraints. There are some rules related to the upper and lower limits of the independent variables that must be observed in the process, as explained below. To demonstrate this, we use one of the test problems for a 7 x 7 transportation matrix with the sums of rows and columns printed in Table 7.1.

Table 7.1 Sample Transportation Problem

	20	20	20	23	26	25	26
27	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}
28	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}
25	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	X_{36}	X_{37}
20	X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	X_{46}	X_{47}
20	X_{51}	X_{52}	X_{53}	X_{54}	X_{55}	X_{56}	X_{57}
20	X_{61}	X_{62}	X_{63}	X_{64}	X_{65}	X_{66}	X_{67}
20	X_{71}	X_{72}	X_{73}	X_{74}	X_{75}	X_{76}	X_{77}

In this problem the total number of variables is 49, with (7+7-1) = 13 dependent and 36 independent. One can therefore pick any 36 variables as independent. The easiest choice

is to simply pick one row and one column and designate all variables on the chosen row and column as the dependent ones. For example, assume that the last (seventh) row and the last column contain the dependent variables, while the first six rows and columns contain independent variables. Therefore, the first 36 variables (x_{11} through x_{66}) can be picked up in a random order and each of them can be assigned a value. Each time a value is allocated to one of the independent variables, the current sum of the total allocated flow in the corresponding row and column is updated, along with the total sum of flow of all independent variables. Rather than allocating values to independent variables in a completely random manner, a more tuned approach can be used based on the knowledge of the objective function. It is possible to inspect objective function for each decision variable prior to starting the initialization procedure and save the knowledge about its local minimums, which can then be used at the time an independent variable is allocated a value. This process is defined by calling a value from the function *allocate* on line 10 of the pseudo code shown below, and it will be addressed again later.

```
      Procedure Initialization
 1)   set x_ij = 0              i = 1, m-1;  j = 1, n-1;
 2)   set sumx = 0
 3)   set sumrow(i) = s(i)          i = 1, m-1;
 4)   set sumcolumn(j) = d(j)       j = 1, n-1;
 5)   set minimum_sum = uniform(minflow, maxflow)
 6)      while( sumx <=  minimum_sum)
 7)         j = uniform( 1, m-1 )
 8)         i = uniform( 1, n-1 )
 9)         limit = minimum(sumrow(i), sumcolumn(j))
10)         x_ij = x_ij + min(limit, allocate(sumrow(i), sumcolumn(j))
11)         sumrow(i) = sumrow(i) - x_ij
12)         sumcolumn(j) = sumcolumn(j) - x_ij
13)         sumx = sumx + x_ij
14)      end
      end
```

The above procedure starts on line 1) by initializing all independent variables to zero. On line 2) variable sumx, which represents the sum of all independent variables is set to zero while variables sumrow(i) and sumcolumn(j) are set on lines 3) and 4), respectively, to the initial values given in Table 7.1 for each row and column. Variable minimum_sum represents the sum of all independent variables. It is set to a randomly chosen value between the previously set minflow and maxflow values. Variable sumx, which represents the total sum of all independent variables, must be within a range defined by minflow and maxflow. The definition of minflow and maxflow is given more attention in the following.

Consider the values of sums and rows given in Table 7.1. The total flow through the entire system including both dependent and independent variables is 160. This can be calculated by summing either all column sums (i.e. 20+20+20+23+26+25+26) or by summing the row sums, which, due to the definition of the balanced transportation problem, must be equal. If the seventh column and the seventh row have been chosen as the dependent variables, the maximum flow that can ever be allocated to the dependent variables is 26+20 = 46. This happens in the case when a feasible solution contains variable x_{77} which equals zero. In that case, the minimum flow that must be allocated to independent variables is equal to 160 - 46 which is 114. On the other hand, when variable x_{77} has its maximum value, which is 20, the sum of flows along the dependant row and the dependent column is 26. In that case, the minimum total flow that must be allocated to the independent variables is equal to 160-26, which is 134. Hence minflow is 114 and maxflow is 134. These considerations result from the maximum flow -- minimum cut theorem from network flow theory. Any transportation problem can be viewed as a network flow problem, and the variables in a particular row or column define a cut - a set of arcs which isolates a given set of nodes (source or destination) from the other nodes in the network. To generalize, variables minflow and maxflow above are defined as follows:

$$\text{minflow} = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} - \max\left\{\sum_{i=1}^{m} x_{in} + \sum_{j=1}^{n} x_{mj} - x_{mn}\right\} \tag{7.3}$$

$$\text{maxflow} = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} - \min\left\{\sum_{i=1}^{m} x_{in} + \sum_{j=1}^{n} x_{mj} - x_{mn}\right\} \tag{7.4}$$

Setting the variable minimum_sum in the pseudo code to a randomly chosen value between 114 and 134 was done to ensure that all corners of the feasible region have equal chances of being addressed. The body of the while loop between lines 6) and 14) represents the initialization process. On lines 7) and 8) row index i and column index j are picked randomly, on line 9) variable *limit* is assigned a value using the minimum of the currently remaining capacity of the corresponding row and column, on line 10) independent variable x_{ij} is allocated a value using the *allocate* function mentioned above (assignment operator is used since one x_{ij} may be visited more than once in a single initialization). Finally, on lines 11), 12) and 13) variables sumrow(i), sumcolumn(j) and sumx are updated. The process goes on until variable sumx becomes greater or equal to the specified minimum_sum.

The process is finalized by solving the dependent variables on the seventh row and the seventh column, which ensures feasibility (not shown in the pseudo code above). This procedure is repeated 500 to 1000 times to create the initial population. Each individual has the variable minimum_sum set randomly to a different value between 114 and 134, to ensure that all corners of the feasible region receive equal attention.

The *allocate* function could have been a simple uniform guess between zero and the current limit set on line 9). However a simple uniform guess is often not very intelligent. For example, in most transportation problems the goal is to minimize the total cost of shipment, so many variables in the final solution are set to zero, while a handful of others have high values. It is easy to see that the chances of guessing a zero with a uniform distribution between zero and a positive number are virtually nil. Therefore, by inspecting the shape of the cost function for each decision variable at the outset, a much better guess can be made regarding its value, with a higher likelihood of hitting the corners of the feasible region which are essential for accelerating further search. In other words, the use of simple heuristic

rules can significantly increase the chances of generating some good solutions in the initial population. This can lead to major shortcuts in the rest of the search process. The shapes of the objective functions chosen for the test problems demonstrate this clearly, and this issue will be revisited in the following sections.

Another important observation should be made at this point: given a large number of independent variables, it is possible that the initialization process could accommodate some functional relationships between them. For example, some of the variables could be completely dependent on the others, or perhaps their upper limit may be a function of the values of other independent variables. This would require a small adjustment in the initialization procedure to generate the independent variables first, recalculate their dependent counterparts, generate the remaining independent variables which have no dependent counterparts, and then recalculate the values in the dependant row and column. Similar observation can be made for mixed integer problems: integer variables could be assigned only integer values in the initialization procedure. While mixed integer problems with non-linear constraints have not been further explored in this paper, it is worth noting that the proposed initialization procedure and the rest of the algorithm presented here is fully capable of handling them.

It was initially felt that simulated annealing (Rudolph, 1994) could be used to direct the search, in combination with the above procedure. One could obtain the mean and the standard deviation of each decision variable from the chosen sample of the best solutions. However, no clear conclusion regarding favorable search directions could be drawn from the best individuals in the initial population. The approach was tested but convergence was rather slow and the quality of final solutions was inferior to those found in earlier studies (Michalewicz, 1994). The use of a GA based recombination operator has proven to be superior to simulated annealing in this study. However, it will be shown that a combination of the two approaches can be productive in some cases, especially in the final phases of the search.

7.5 Evaluation

Evaluation of the initial population involves calculation of the objective function (fitness value) for each of them. It is a standard step in all GA and EP applications. There are no added penalty functions here since the initialization procedure guarantees that all individuals in the initial population are feasible.

7.6 Selection

A small fraction of the individuals with the best fitness value are selected to join the mating pool. The other individuals from the initial population are of no further interest in the process, they are considered as unsuccessful parents that died without offspring. Various sizes of mating pool were tested and the best results on the test problems used in this study were achieved with the mating pool containing between 15 and 25 individuals. Sorting of the entire initial population is not used, since the selection process does not require that all individuals in the initial population be ranked. What is needed is merely a selection of a small fraction of the best.

7.7 Recombination

The recombination operator is the heart of the solution procedure. It is modeled after the natural process known in biology as crossover, which involves two individuals (or genotypes) creating a new organism through sexual reproduction by passing some randomly chosen genetic material from one parent and some from the other.

In technical problems the genetic material is usually represented as a string or an array of numbers which are values of decision variables forming the feasible solution selected for mating. The usual procedure in technical applications of GAs is to break the solution string in only one or two points and conduct a mutual replacement by exchanging the partial string

segments. For example, two parents a_n and b_n would have their solution strings broken at the same point i (where $1 \leq a_i \leq n$). By exchanging the sub-strings, two children would be created (a_i, b_k) and (b_i, a_k) where $k = i+1, n$. The problem in technical applications is that such an operator would often violate the initial feasibility of the parents, since there is no mechanism to preserve feasibility of the offspring. This is why most GA applications with floating point domain representation usually resort to some type of linear combination of parents V_1 and V_2 such as, for example

$$U = c\ V_1 + (1 - c)\ V_2 \tag{7.5}$$

where c is a uniform random number between 0 and 1. This approach guarantees that children of feasible parents are also feasible. Similar linear combinations of parents are used in other search methods, e.g. Scatter Search (Glover, 1999). However, the downside is that such operators can be applied only on problems with linear constraints. The GENETIC-2 model (Michalewicz et al., 1994) uses a variant of this operator.

In nature, genetic material is a long DNA chain which is broken randomly at many points during cross-over. This approach is also based on the multipoint crossover, with the number of points being equal to the number of the decision variables. All relevant information which describes the mating pool was stored in a matrix consisting of 25 rows, each row representing a parent, and 37 columns, with the first column containing the value of the objective function and the remaining 36 containing the values of the independent variables (genes). The crossover technique is based on allowing each of the chosen 25 parents to contribute their genetic material to offspring with equal likelihood. In other words, a new individual is created by selecting randomly a value from each column of the matrix which represents the mating pool, and placing it in the corresponding column of the newly created offspring. In the genetic makeup of a new individual, some parents may contribute more of their genetic material than the others, but this is completely left to chance -- no bias is introduced between the parents based on the differences in their objective function. This was

done since it was felt that ranking such a small selection out of a massive initial population has no justifiable merit. All chosen solutions have some good qualities in them, and with this approach they are allowed to come to the fore without any bias based on relatively small differences in the objective function, which may be caused by only a few "bad" values chosen for some decision variables. In this sense the approach presented here differs from the commonly accepted wisdom that relies on some form of bias among the selected parents, allowing the best parents to pass their genetic material to offspring more often.

The above recombination operator may result in a new choice of independent variables which violate feasibility. Either the sum of an individual row or column may be above the prescribed target, or the sum of all independent variables may be less than the required variable minimum_sum. The fix must therefore be provided as soon as such a condition is encountered, hence the recombination procedure is tied together with the "gene therapy" procedure that fixes defective genes in order to preserve feasibility, as outlined in the next section. Gene therapy procedure also plays a role of a mutation operator, since it modifies the individual values in the mating pool. We summarize the features of the process addressed so far:

- The number of crossover points is the same as the number of independent variables (in this example 36);
- As soon as a new individual is created which has a better fitness value than that of the worst parent in the mating pool, the new individual joins the mating pool, while the worst parent is discarded from it; and,
- Identical twins are not allowed in the mating pool. They tend to reproduce each other and eventually fill the entire mating pool. While this happens gradually and the identical twins represent solutions that are usually of very high quality, they often represent local optima, which should be avoided. **Variety** of good genetic material and **gradual** improvement of its quality is essential to progress towards the best points in the search space.

7.8 Gene Therapy

There are two possible violations of feasibility of the new individual created by recombination. Either the sum of individual rows or columns may be exceeded, in which case a quick reduction to the maximum possible value is required, or the sum of all independent variables may be insufficient, resulting in calling of the initialization procedure to randomly add additional flows to independent variables. To achieve the two fixes in an easy and efficient manner, the entire mating process is carried out under the umbrella of a procedure similar to initialization, which makes sure that the feasibility requirements associated with independent variables are maintained. The gene therapy is therefore a *monitoring and adjustment procedure* which quickly fixes any individual violations of feasibility that may occur in the mating process. The following describes the recombination and the gene therapy procedure:

```
     Procedure Recombination
 1)  set x_ij = 0        i = 1, m-1; j = 1, n-1;
 2)  set sumx = 0
 3)  set sumrow(i) = s(i)          i = 1, m-1;
 4)  set sumcolumn(j) = d(j)       j = 1, n-1;
 5)  set minimum_sum
 6)  generate_selection
 7)  while( there is at least one unvisited gene)
 8)        j = pick_from_selection
 9)        i = pick_from_selection
10)        limit = minimum(s(i), d(j))
11)        p = uniform(1, 25)
12)        x_ij = xbest_pij
13)        x_ij = minumum(x_ij, limit)
14)        sumrow(i) = sumrow(i) - x_ij
15)        sumcolumn(j) = sumcolumn(j) - x_ij
16)        sumx = sumx + x_ij
17)  end
```

```
17)   if(sumx < minimum_sum)
18)         procedure initialization
19)   end
```

Lines 1) through 4) are the same as in the initialization procedure. Line 5) differs a bit, since the initial mating pool provides sufficient information from which a desirable range for the sumx variable can be selected more accurately, i.e. a better informed guess can be made based on the properties of the already chosen for reproduction than a mere uniform guess between the minflow and maxflow values. On line 6) a random sequence of numbers from 1 to 36 is created to ensure that the sequence of allocating genes to the new individual is random. Lines 7) through 17) capture the process, in which each gene (decision variable) is allocated from a participating parent which is randomly chosen on line 11), with the actual allocation of genetic material carried out on line 12). Gene therapy is performed only if required on line 13) and on lines 18) and 19).

The mating pool is represented by a three dimensional array $xbest_{pij}$, where p=(1, 25) is the parent index while *i* and *j* are the row and column indices representing independent variables. After finishing one pass of the above procedure, it is still necessary to re-calculate the dependent variables and recalculate the objective function of the new individual. The new selection then proceeds immediately for each individual. If the individual has a better fitness value than the worst parent in the mating pool, it will be placed in the mating pool in its appropriate position, pushing all parents with less favorable fitness down by one place and pushing the worst parent out of the mating pool. Incest is allowed in this model (i.e. parents mate with children) and in general all parents produce a new individual. The best parents may survive to mate with many future generations. One could talk about a whole generation with a single individual (Hunter, 1998), although there is not much point being strict about the use of the term "generation" any more. Total lifetime of one individual is only a function of its fitness value and the fitness value of other individuals in the mating pool. If the best individual is found by chance in the initialization procedure, it will outlive all of its offspring.

Calling the initialization procedure at the end of recombination is usually not necessary, but even when it happens it is only executed for a few randomly selected independent variables which have their flows marginally increased. This serves a twofold purpose: it ensures feasibility, and it also serves as a mutation operator since it adds a small variation to the genetic makeup of the parents. A total of 10 test problems were solved using the above approach, and only two of them required modification to this algorithm by introducing an additional mutational operator. This is further addressed in the following section.

To summarize, the heart of the algorithm is the recombination procedure, the selection procedure and the process of updating the membership of the mating pool. The entire algorithm is depicted below:

```
     Procedure main
 1)  initialization
 2)  evaluation
 3)  selection
 4)        while ( terminating condition not true )
 5)              recombination
 6)              evaluation
 7)              if (fitness(x_ij ) < fitness (xbest_25,i,j))
 8)                    update xbest_pij
10)        end
11)  end
```

On line 7) a the fitness of the new individual is compared with the fitness of the worst parent. If better, the mating pool is updated. If not, a new individual is created. The total number of generated solutions x_{ij} was used as a terminating condition in the test problems in this research. Two out of five problems tested with 49 variables already converged after 3000 individuals were generated (initial population of 500 and a total of 2500 individuals created as a result of mating). This is encouraging considering that the search space consists of 49 floating point variables.

It should be noted that the gene therapy is also capable of introducing adjustments that may be required to preserve the mixed integer nature of the problem, or to preserve the non-linear relationships in the constraints.

7.9 Similarity to Minimum Cost Network Flow Problems

Both transportation problems and minimum cost network flow problems share similar constraints, associated with (a) minimum and maximum flow along an arc; (b) minimum and maximum flow through a node; and (c) a continuity equation for each node. Of those, only minor adjustment of the algorithm presented here is needed to include constraint (a). Selection of one dependent column and one row in the TP is equivalent to making a selection of arcs that form a maximum spanning tree in a network, which define a set of dependent flow variables. Flows on arcs which do not belong to the maximum spanning tree are the independent variables in network flow problems (Ahuja et al., 1993).

7.10 Test Problems

The objective function for both sets of test problems (7 x 7 and 10 x10) is of the form:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \cdot f(x_{ij}) \tag{7.6}$$

The shape of the objective function f is the same on all arcs. The variation between arcs is achieved with the cost parameter c_{ij}. The 7 x 7 problem is defined below:

Source Flows:	27	28	25	20	20	20	20
Destinations:	20	20	20	23	26	25	26
Cost c_{ij}:	0	21	50	62	93	77	1000
	21	0	17	54	67	1000	48
	50	17	0	60	98	67	25
	62	54	60	0	27	1000	38
	93	67	98	27	0	47	42
	77	1000	67	1000	47	0	35
	1000	48	25	38	42	35	0

Note that the diagonal elements have zero cost parameters, while there are six cost parameters with very large value of 1000 in relative comparison to the rest. The 10 x 10 cost matrix and source/destination flows are shown below:

Source Flows:	8	8	2	26	12	1	6	18	18	1
Destinations:	19	2	33	5	11	11	2	14	2	1
Cost c_{ij}:	15	3	23	1	19	14	6	16	41	33
	13	17	30	36	20	17	26	19	3	33
	37	17	30	5	48	27	8	25	36	21
	13	13	31	7	35	11	20	41	34	3
	31	24	8	30	28	33	2	8	1	8
	32	36	12	9	18	1	44	49	11	11
	49	6	17	0	42	45	22	9	10	47
	2	21	18	40	47	27	27	40	19	42
	13	16	25	21	19	0	32	20	32	35
	23	42	2	0	9	30	5	29	31	29

The above problems were taken from the literature since they were tested earlier (Michalewicz et al., 1991; Michalewitz, 1994), with a total of six objective functions with the best results being produced by the GENETIC-2 model. Those functions plus a standard LP form were chosen for this study as defined below and labeled in the same way as in the work of Michalewicz et al.

Function A:
$$f(x) = \begin{cases} 0 & \text{if } 0 < x_{ij} \le 2; \\ c_{ij} & \text{if } 2 < x_{ij} \le 4; \\ 2c_{ij} & \text{if } 4 < x_{ij} \le 6; \\ 3c_{ij} & \text{if } 6 < x_{ij} \le 8; \\ 4c_{ij} & \text{if } 8 < x_{ij} \le 10; \\ 5c_{ij} & \text{if } 10 < x_{ij} \end{cases} \tag{7.7}$$

Function B:
$$\begin{cases} 0.2\, c_{ij}\, x_{ij} & \text{if } 0 < x_{ij} \le 5; \\ c_{ij} & \text{if } 5 < x_{ij} \le 10; \end{cases} \tag{7.8}$$

c_{ij} (1 + 0.2(x_{ij} − 10)) if 10 < x_{ij} ;

Function C: $c_{ij}\, x_{ij}^2$ (7.9)

Function D: $c_{ij}\sqrt{x_{ij}}$ (7.10)

Function E: $$c_{ij}\left[\frac{1}{1+(x_{ij}-10)^2}+\frac{1}{1+(x_{ij}-11.25)^2}+\frac{1}{1+(x_{ij}-8.75)^2}\right]$$ (7.11)

Function F: $$c_{ij}x_{ij}\left|\sin\left(x_{ij}\frac{\pi}{4}\right)+1\right|$$ (7.12)

Function G: $c_{ij}x_{ij}$ (linear programming case) (7.13)

Function G adds a linear programming test in addition to the basic tests conducted earlier. Before discussing any individual results, it is useful to look at the shape of each function depicted in Figures 7.1 through 7.6. Simple heuristic rules have been applied in each of the above functions during the initialization procedure to increase the likelihood of having parents with high fitness values in the initial mating pool. The knowledge of the shape of the objective function was used in the process of building the heuristics. The hardest functions to solve were C and F. The results are first discussed for functions A, E and D. It may be mentioned at this point that GENETIC-2 experiments were conducted with 35000 individuals generated for the 7 x 7 problem and 50000 individuals generated for the 10 x 10 problem.

Figure 7.1 Function A

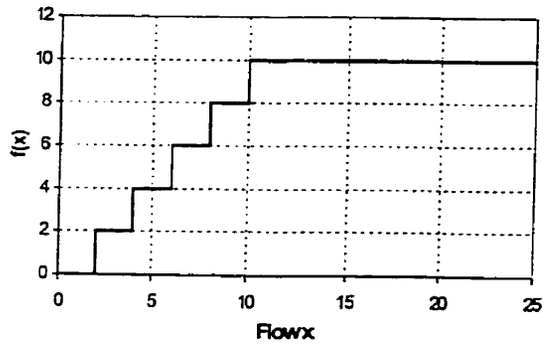


Figure 7.2 Function B

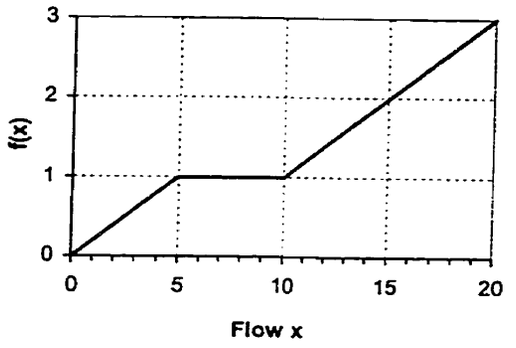


Figure 7.3 Function C

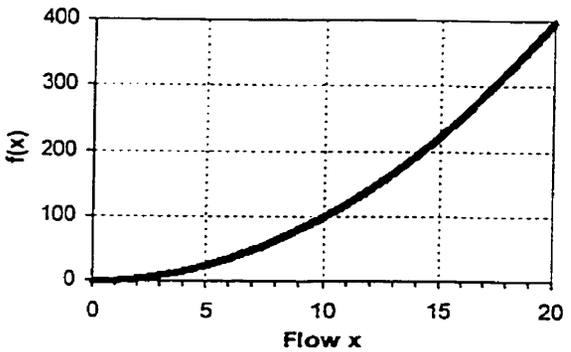


Figure 7.4 Function D

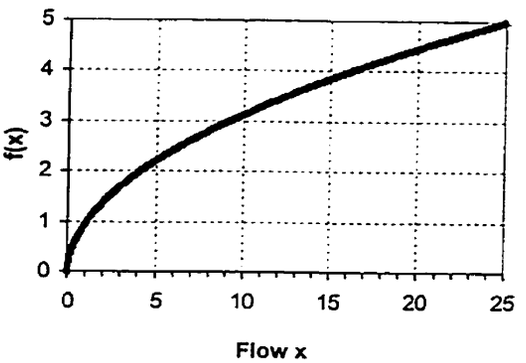


Figure 7.5 Function E

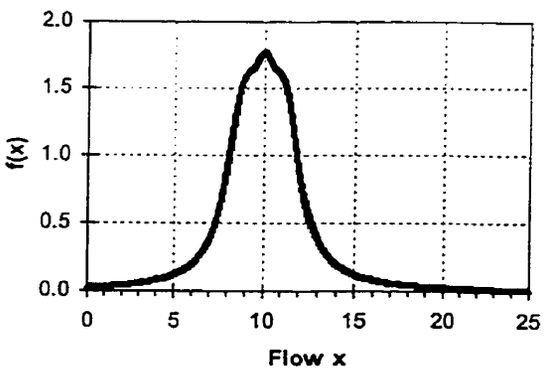
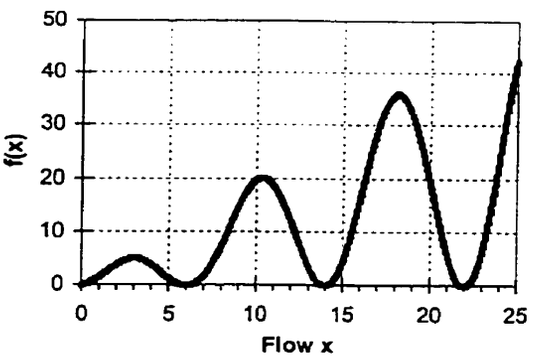


Figure 7.6 Function F



7.10.1 Function A

It is obvious from the shape of this function that flows on any arc that are less than 2 would result in a zero cost of flow along such an arc. The initialization procedure was therefore modified to increase the likelihood that the decision variables with $c_{ij} > 0$ set to 2, while the decision variables with $c_{ij} = 0$ were allowed to have higher values. Many initial solutions that could not comply with these heuristic rules had to be allowed due to random sequence of allocating values to x_{ij}. Strict adherence to this rule precludes the sum of all independent variables to equal the required minimum sum, given the random sequence of allocating values to decision variables.

For the 7 x 7 problem the mating pool was selected from the initial 500 individuals, with the objective function of the chosen parents ranging from 83 to around 200. After 4500 new individuals were created the mating pool had the values of the objective function ranging from 0 to 65. The best 7 parents had their objective function equal to zero. Four of them are shown in Table 7.2.

Table 7.2 Solutions for 7 x 7 Problem with Function A

19.69	0.15	0.53	1.85	2.00	1.03	1.74	20.00	0.00	1.38	0.92	2.00	1.91	0.78
0.00	19.85	1.12	1.93	2.00	2.00	1.10	0.00	20.00	0.88	1.93	1.85	1.95	1.38
0.31	0.00	17.60	2.00	2.00	2.00	1.10	0.00	0.00	17.73	2.00	1.81	1.90	1.56
0.00	0.00	0.00	17.21	0.53	1.87	0.39	0.00	0.00	0.00	18.15	0.00	1.12	0.73
0.00	0.00	0.75	0.00	19.25	0.00	0.00	0.00	0.00	0.00	0.00	19.89	0.11	0.00
0.00	0.00	0.00	0.00	0.22	18.10	1.67	0.00	0.00	0.00	0.00	0.45	18.00	1.55
0.00	0.00	0.00	0.00	0.00	0.00	20.00	0.00	0.00	0.00	0.00	0.00	0.00	20.00

20.00	0.00	1.15	1.98	2.00	0.05	1.81	20.00	0.17	0.00	0.92	2.00	2.00	1.91
0.00	20.00	1.55	1.93	2.00	2.00	0.52	0.00	19.83	1.61	1.95	2.00	2.00	0.62
0.00	0.00	17.30	2.00	2.00	1.97	1.73	0.00	0.00	18.39	2.00	1.65	2.00	0.95
0.00	0.00	0.00	17.09	0.00	1.00	1.91	0.00	0.00	0.00	18.13	0.46	1.00	0.41
0.00	0.00	0.00	0.00	19.97	0.00	0.02	0.00	0.00	0.00	0.00	19.89	0.00	0.11
0.00	0.00	0.00	0.00	0.03	19.97	0.00	0.00	0.00	0.00	0.00	0.00	18.00	2.00
0.00	0.00	0.00	0.00	0.00	0.00	20.00	0.00	0.00	0.00	0.00	0.00	0.00	20.00

The power of the initialization procedure becomes more evident in the case of the 10 x 10 problem. The initialization procedure generated 1000 individuals. The best among them had a fitness value of 216, which is fairly close the optimum of 202 found by GENETIC-2 and much better than the optimum of 281 found with the GAMS (Brooke et al., 1996) solver, as reported by Michalewicz et al. After 10,000 additional individuals were generated, all members of the mating pool had their fitness value below 201, with the best of them shown in Table 7.3 equal to 173.

Table 7.3 Solution for 10 x 10 Problem with Function A

1.81	0.68	1.82	0.00	1.21	0.00	0.49	2.00	0.00	0.00
0.96	0.00	2.00	1.99	2.00	0.00	0.00	0.77	0.28	0.00
0.00	0.00	0.06	0.00	0.00	0.00	0.36	1.57	0.00	0.00
0.43	0.00	20.38	0.59	2.00	0.00	0.00	2.00	0.59	0.00
2.00	1.13	3.53	1.46	2.00	0.00	0.00	1.88	0.00	0.00
0.99	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
0.00	0.19	1.66	0.00	1.06	0.54	0.00	1.85	0.77	0.00
11.53	0.00	1.54	0.24	1.15	0.88	0.73	1.92	0.00	0.00
1.28	0.00	2.00	0.72	1.63	9.58	0.42	2.00	0.36	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

7.10.2 Function B

Function B is very similar to a linear function. Judging by the shape of this function, the obvious heuristic rule here would be to favor allocations of less than 10 on any arc with a non-zero cost. Allocation on arcs with zero cost could be maximized if necessary, since it does not affect the objective function. Table 7.4 shows the solution for the 7 x 7 problem. The objective function value is 203.75, which is almost the same as 203.81 obtained earlier by Michalewicz et al. This solution was obtained within 5000 iterations. Table 7.5 shows the final solution for function B with 10 x 10 matrix. This solution was obtained after 10000 iterations, with objective function of 159.79, which is a 2% improvement over 163.0 reported earlier.

Table 7.4 Solution for 7 x 7 Problem with Function B

20.00	0.16	0.00	2.35	0.00	4.49	0.00
0.00	19.83	1.01	0.17	6.98	0.00	0.00
0.00	0.00	18.99	0.00	0.00	0.00	6.00
0.00	0.00	0.00	20.00	0.00	0.00	0.00
0.00	0.00	0.00	0.48	19.02	0.51	0.00
0.00	0.00	0.00	0.00	0.00	20.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	20.00

Table 7.5 Solution for 10 x 10 Problem with Function B

0.00	1.56	1.33	2.71	1.25	0.02	0.70	0.42	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.77	0.23	0.00
0.00	0.00	0.05	0.65	0.00	0.00	1.30	0.00	0.00	0.00
11.15	0.44	10.03	1.65	0.00	2.73	0.00	0.00	0.00	0.00
0.00	0.00	10.24	0.00	0.00	0.00	0.00	0.00	1.76	0.00
0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.18	0.00	0.00	0.00	0.00	5.81	0.01	0.00
7.84	0.00	10.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	9.75	8.25	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

7.10.3 Function E

This function is yet another case in favor of modifying the initialization procedure such that parents with good fitness values can be generated in the initial population. It is obvious from the shape of the objective function that the most favorable values are either zero or over 20. In fact, the highest the value of flow along an arc, the better, since the objective function has a small positive value of about 0.03 even for $x_{ij} = 0$. The initialization procedure was modified in the following way. Instead of:

$$x_{ij} = x_{ij} + \text{allocate}(\, i, j \,) \tag{7.14}$$

the model used

$$x_{ij} = \text{limit} \tag{7.15}$$

This approach was chosen to demonstrate one more feature inherent in this algorithm. Knowing that the constraints (sums of rows and columns) are all integers, this initialization procedure with the above modification will result in the search space of integer solutions only. While the global optimum may not be an all-integer solution, the best integer solutions can be found very quickly. Added search in the floating point space is possible by introducing an additional mutation operator, which is done for functions C and F, but not here. The best solution found for a 7 x 7 problem has fitness value of 204.88, fairly close to 204.73 found by GENETIC-2. However, the fitness values of the best solutions found for 10 x 10 problem are all within 71.83 and 72.21, almost 10% better than 79.2 found by GENETIC-2. It took less than 5000 individuals to reach these solutions for 10 x 10 problem, and less than 3000 for 7 x 7 problem. Table 7.6 displays the best solution for 10 x 10 problem with objective function equal to 71.83.

Table 7.6 Solution for 10 x 10 Problem with Function E

0.00	1.00	0.00	2.00	3.00	0.00	0.00	2.00	0.00	0.00
1.00	1.00	1.00	0.00	2.00	0.00	0.00	2.00	1.00	0.00
0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00	0.00	0.00
0.00	0.00	26.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	3.00	0.00	1.00	0.00	2.00	4.00	1.00	1.00
0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	1.00	2.00	0.00	0.00	0.00	3.00	0.00	0.00
17.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.00	0.00	1.00	0.00	3.00	11.00	0.00	2.00	0.00	0.00
0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00

Although only integer solutions were inspected in both the 7 x 7 and the 10 x 10 test problems, it was found that large differences of the argument values among the best individual solutions for the 10 x 10 problem result in very small differences in the objective function, which demonstrates the existence of numerous local optima. This offers a hint as

to why simulated annealing approach, which was initially tried in a combination with the initialization procedure, had so much difficulty converging to high quality solutions.

7.10.4 Function D

This is also a function which gives large contribution to the overall cost even for small values of decision variables. For example, for a decision variable equal to 0.2 the square root is 0.45, which, when multiplied with a cost factor of 1000 gives 45. One conclusion that can be drawn out of this is regarding the 7 x 7 problem is that decision variables with a cost parameter set to 1000 should ideally be set to zero. This heuristic rule was built in the 7 x 7 problem. Other than that, the shape of the function reveals that several large values of the decision variables in combination with many zeros would most likely result in the best overall solution. Hence, the initialization procedure was modified as follows.

$$\text{allocate}(i,j) = \text{minimum}\{[\text{limit} + \text{limit} \cdot N(0,1)], \text{limit})\} \tag{7.16}$$

where N(0,1) is a normal variate with a mean of zero and standard deviation of 1. Hence, there is a 50% chance that a decision variable is set to its current maximum, and a 50% that it would be less than maximum, but it still remains in the high range with respect to the constraints.

For the 7 x 7 test problem, the model found a solution with the same fitness value of 480.16 as the one found by GENETIC-2. For the 10 x 10 problem, there was a very small improvement to the solution of 391.9 obtained by GENETIC-2. The fitness value is 388.91 and the solution is shown in Table 7.7.

Table 7.7 Solution for the 10 x 10 Problem with Function D

1.00	2.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	3.00	0.00	0.00	3.00	2.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00
0.00	0.00	25.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
0.00	0.00	7.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.00	0.00	0.00
18.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	8.00	10.00	0.00	0.00	0.00	0.00
0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

7.10.5 Function C

This function is a textbook example of a regular quadratic program for which GAMS and other gradient based solvers should easily find the global optimum. This problem does not have a well defined favorable corners of the feasible region that could be exploited in the initialization procedure. There are however some simple rules that could be used: decision variables with very large costs (e.g. 1000) should be kept equal to zero if possible, decision variables with cost parameters equal to zero should be allowed to have high values, and the remaining decision variables should have values of around 1 or less, since the quadratic function is exponentially decreased for the argument less than 1. The main difficulty with this function was a tendency of the algorithm to cluster around a local optimum, since all members of the mating poll begin to look like identical twins, but not quite since the numbers differ on the second decimal. So there was a phenomena of having "almost identical twins" with negative impacts on convergence, the same problem encountered with the emergence of identical twins. To make sure the variety of good genetic material is retained in the mating pool, a small mutation operator was added in the recombination procedure in the following form:

$$x_{ij} = \text{xbest}_{pij} + \theta \, \text{xbest}_{pij} \, N(0,1) \tag{7.17}$$

where θ was tested with values between 0.1 and 0.01, with 0.05 being the best value after conducting several test runs. In other words, the parent's gene was copied but it was mutated slightly by taking 5 percent of its value as the mean of the standard deviate. The algorithm converged to good solutions, but it took more computational effort than in the case of the previous three test functions. This approach represents a form of a marriage of GA based recombination with simulated annealing. More avenues are being explored at present in order to ensure a more efficient solution procedure for this type of function.

Typically, the gradient search methods (in this case quadratic programming) are expected to find a global optimum for these objective functions, and other heuristic techniques can be gauged by measuring their closeness to the global optimum found by a quadratic programming model. Indeed, the entire explosion of interest in heuristic methods is not because of problems of this kind, but rather the problems with objective function of type E or in particular F, where the gradient search methods fair poorly in comparison to GA methods. Table 7.8 shows the solution obtained from the GAMS solver, with the objective function equal to 2535.29, and the best solution from the SFEP after 20000 individuals, with the objective function equal to 2534.34.

Table 7.8 Solution of 7 x 7 Problem with Function C

GAMS Solver							SFEP						
20.00	0.52	0.85	1.83	1.59	2.08	0.14	19.83	0.46	1.07	1.49	1.67	2.29	0.19
0.00	19.48	1.86	1.89	2.04	0.15	2.59	0.13	19.51	1.62	2.23	2.09	0.00	2.11
0.00	0.00	17.29	1.18	1.07	1.75	3.70	0.00	0.01	17.31	1.09	0.99	1.83	3.77
0.00	0.00	0.00	18.10	1.27	0.05	0.58	0.03	0.01	0.00	18.19	1.36	0.00	0.39
0.00	0.00	0.00	0.00	19.74	0.26	0.00	0.00	0.00	0.00	0.00	19.51	0.24	0.25
0.00	0.00	0.00	0.00	0.00	20.00	0.00	0.00	0.00	0.00	0.00	0.08	19.91	0.00
0.00	0.00	0.00	0.00	0.29	0.71	19.00	0.00	0.00	0.00	0.00	0.30	0.42	19.28

While their fitness value is almost the same, some of the individual decision variables differ by over 20%. There is room for further improvement in the efficiency of SFEP search based on the ideas outline above.

7.10.6 Function F

This is a challenging function with four possible roots for argument values of 0, 6, 14 and 22. The initialization procedure was adjusted in such a way as to favor any of those values. The same mutation operator as the one described for function C was applied here as well. Convergence is slower, however the solutions for both problems are better than those found by GENETIC-2. For the 7 x 7 problem, the entire mating pool ends with the fitness values between 78.81 and 83.18 while the best solution previously found with GENETIC-2 had a fitness value of 110.94. Table 7.9 shows the two solutions from the mating pool with their objective function values, found after 20000 individuals were generated in the process. The 10 x 10 problem was run for 40000 individuals. The best individuals generated in the process had their fitness values ranging from 173.26 to 175.45 while the best solution previously found by GENETIC-2 had a fitness value of 201.9. Table 7.10 shows the best solution generated for the 10 x10 test problem. It should be noted that no attempt was made to optimize the efficiency of the search process. Examined at this point were only the robustness of the approach and its overall capability to find good solutions.

Table 7.9 Two solutions for 7 x 7 Problem with Function F

Objective Function = 78.84							Objective Function = 79.72						
14.20	0.54	0.08	0.09	6.11	5.97	0.00	14.28	0.48	0.03	0.13	6.11	5.97	0.00
0.00	7.99	5.94	14.06	0.00	0.00	0.00	0.00	7.96	0.00	14.04	0.00	0.00	6.00
5.80	5.58	1.86	0.00	6.02	5.75	0.00	5.71	5.59	1.90	0.00	5.93	0.00	5.85
0.00	5.86	5.90	2.46	5.77	0.00	0.00	0.00	5.97	5.88	2.41	5.74	0.00	0.00
0.00	0.00	0.00	0.14	8.06	5.73	6.06	0.00	0.00	0.00	0.14	8.09	5.73	6.03
0.00	0.00	6.21	0.00	0.03	7.54	6.21	0.00	0.00	6.17	0.00	0.10	7.54	6.18
0.00	0.02	0.00	6.24	0.00	0.00	13.72	0.00	0.00	6.00	6.27	0.03	5.76	1.94

Table 7.10 Solution of 10 x 10 Problem with Function F

0.25	1.43	0.01	0.00	5.21	0.00	0.42	0.64	0.04	0.00
0.00	0.00	0.00	0.00	0.00	5.79	0.00	0.25	1.96	0.00
0.00	0.38	0.00	0.00	0.00	0.00	1.27	0.31	0.00	0.04
0.00	0.00	14.07	4.96	0.00	0.00	0.00	6.15	0.00	0.81
6.04	0.00	5.54	0.00	0.00	0.00	0.20	0.22	0.00	0.00
0.07	0.03	0.61	0.00	0.05	0.09	0.00	0.00	0.00	0.15
0.00	0.00	5.85	0.03	0.02	0.00	0.00	0.08	0.00	0.00
6.24	0.03	6.02	0.00	5.71	0.00	0.00	0.00	0.00	0.00
6.40	0.12	0.00	0.00	0.00	5.12	0.00	6.35	0.00	0.00
0.00	0.00	0.88	0.00	0.00	0.00	0.11	0.00	0.00	0.00

7.10.7 Function G

Pure linear programming case has been added to the set of test runs above. It is easy to solve using available LP solvers and it provides a way of measuring the ability of the SFEP solver to find the global optimum. Micorsoft Excel solver was used as LP solver in this case. The 7 x 7 problem was run with 3000 iterations, and 10 x 10 problem was extended to 5000 iterations. The initialization function was modified to include only guesses which are equal to maximum possible flow at any time. In this way the SFEP is encouraged to search only the corners of the feasible region, similar to other LP solvers. Identical solution was reached by SFEP and the Excel solver for the 7 x 7 problem, with the objective function equal to 113.2. This solution is shown in Table 7.11.

Table 7.11 Solution of 7 x 7 Problem with Function G

20.00	0.00	0.00	2.00	0.00	5.00	0.00
0.00	20.00	1.00	1.00	6.00	0.00	0.00
0.00	0.00	19.00	0.00	0.00	0.00	6.00
0.00	0.00	0.00	20.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	20.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	20.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	20.00

Table 7.12 shows the SFEP solution for the 10 x 10 problem. Here the solution is not identical, the SFEP failed to converge to the global optimum by a small margin, with the final objective function of 118.1 while the Excel's LP solver converged to 117.9. The difference of 0.17% may not seem significant, but it does serve as a reminder that SFEP and other similar algorithms provide no proof of converging to a global optimum. After inspecting the values of decision variables in the mating pool, it was obvious that the global optimum values were all there, but their right combination did not emerge after 5000 iterations. Rather than increase the number of iterations, the way to compete with LP solvers on large problems would be to include additional heuristic rules in the search process. These could be related to mutation or recombination operators. One thing should be noted: there is no benefit to applying the SFEP where other available LP solvers are much faster and they guarantee global optimum. A useful comparison between SFEP and an LP solver would be to use a large dense network and modify the SFEP to solve the dual problem. This has double advantage to SFEP: (a) no need for gene therapy, since dual variables are unrestricted in sign and unlimited in value; and, (b) fewer variables, since there are typically fewer nodes than arcs. For example, in the 10 x 10 transportation problem there are 19 dual variables that must be found to retrieve the optimal values of all 100 arc flows. To solve an equivalent (dual) problem with 19 unconstrained variables may be easier than to struggle with 81 independent flow variables which are also heavily constrained.

Table 7.12 Solution of 10 x 10 Problem with Function G

0.00	1.00	0.00	0.00	0.00	0.00	0.00	7.00	0.00	0.00
0.00	0.00	0.00	0.00	5.00	0.00	0.00	1.00	2.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00	0.00	0.00
19.00	1.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	1.00
0.00	0.00	12.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.00	0.00	0.00
0.00	0.00	18.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	1.00	0.00	6.00	11.00	0.00	0.00	0.00	0.00
0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

7.11 Comparison of Results

Table 7.13 provides a comparison of the results from this research with those previously reported by Michalewicz (1994) with two GA programs: GENOCOP and GENETIC-2. Note that GENETIC-2 was specifically designed for transportation problems. Solutions obtained with the GAMS solver were compared with the GENETIC-2 and SFEP programs in Table 7.14.

Table 7.13 Comparisons of Results for the 7 x 7 Test Problems

Function	GENOCOP	GENETIC-2	SFEP	SFEP # Iterations
A	24.15	0	0	5000
B	205.6	203.81	203.75	5000
C	2571.04	2564.23	2534.34	20000
D	480.16	480.16	480.16	3000
E	204.82	204.73	204.88	3000
F	119.61	110.94	78.81	20000

Table 7.14 Comparisons of Results for the 10 x 10 Test Problems

Function	GAMS	GENETIC-2	SFEP	SFEP # Iterations
A	281	202	173	10000
B	180.8	163	159.79	10000
C	4402.04	4556.2	4435.49	40000
D	408.4	391.1	388.91	5000
E	145.1	79.2	71.83	5000
F	1200.8	201.9	173.26	40000

Of the above, the only function where GAMS found a better solution was function C for the 10 x 10 problem, although the difference is only 0.7%. Solutions were obtained quickly. The last column shows the number of iterations needed to find the best solution. Execution

times could not be compared directly with the previous work of Michalewicz et al. since they were never reported earlier due to a variety of computer hardware that was used in their study. The CPU times in this study were timed separately for the 7 x 7 and 10 x 10 problem. To generate 5000 iterations, it takes 0.13 seconds of CPU time for the 7 x 7 problem, and 0.55 seconds for the 10 x 10 problem. The test runs were conducted on a 350Mhz IBM compatible PC with AMD processor. The timing results show that execution times grow exponentially with the size of the problem, which was expected.

7.12 Summary

This Chapter presents an evolution program for solving non-linear transportation problems with possible extension to network flow problems in general. The main features of the solution procedure are:

- Massive initialization procedure which generates only feasible solutions with increased likelihood of generating solutions with good fitness based on simple heuristic rules;
- A multipoint recombination operator which gives all parents in a small mating pool equal chances of contributing their genetic material to offspring;
- An elitist selection operator which places offspring in the mating pool only if its fitness is better than the fitness of the worst parent, resulting in the removal of the worst parent from the mating pool. The selection operator does not allow identical twins in the mating pool.
- A gene therapy operator which fixes defective genes (those that violate feasibility as a result of the genetic crossover), thus ensuring that offspring is always feasible; and,
- Non-linear objective functions and constraints can both be included in the search process.

The procedure seems to offer a good potential to become an efficient solver for a large class of network flow problems with non-linear objective functions and constraints.

8 CASE STUDY II – OPERATION OF BIGHORN/BRAZEAU HYDRO POWER SYSTEM OF TRANSALTA UTILITIES CORPORATION

8.1 Introduction

Finding the best reservoir operating rules is a complex problem, characterized with non-linearly constrained decision variables which vary both in space and in time and have a stochastic component associated with the uncertainty of reservoir inflows. The development of reservoir operating rules has been closely associated with the development of mathematical models to represent the decision making process. There is no universal approach in dealing with this problem. Yeh (1985) provides a summary of various approaches that have been tried in the past. Two groups of methods that have gained general acceptance among researchers are summarized below.

The first group is the *explicitly stochastic* methods (Simonovic 1987, Loucks et al. 1981, Young 1967). These methods require that the entire problem be formulated and solved as a stochastic mathematical program, which means that the uncertainty is represented in some way with a random function imbedded in the model. These methods have suffered from (a) computational inefficiency; (b) difficulties related to their proper mathematical formulation; and, (c) distrust on the part of the reservoir operators due to complexity of the model. Many models have failed to become practical operational tools (Oliviera and Loucks, 1997), even though the final formulation of the reservoir operating rules is usually simple, in the form of tables and graphs (Wurbs, 1996).

The second group is the *deterministic optimization methods*. They are easier to understand and much easier to solve. They rely on known inflows, while the reality is that inflows are unknown.

To overcome this, researchers have resorted to the use of long historic time series of

naturalized inflows, where variability and seasonality of the series includes the necessary stochastic component. The deterministic models could be used in two ways: either (a) develop optimal rules based on the perfect foreknowledge of the inflow series and target releases for a year (here the operating rules given as the draw down and refill curves constitute model output); or, (b) assume an operating rule for a reservoir, run the entire series of historic inflows and current (or projected) demands, and then evaluate the performance of the entire system. In this case the assumed operating rule does not change during the simulation run for all simulated years, and the performance of various scenarios can be compared based on various rules assumed for each scenario.

Approach (a) provides a set of optimal reservoir draw-down and re-fill curves which are different for every year, due to inflows and water demands being also different from year to year. Approach (b) is not so heavily dependent on the inflow series, but it is not easily applicable to multi reservoir systems where the number of possible combinations of rule curve shapes is very large.

Reservoir inflows are an important input for analyses of reservoir operation. Historic hydrologic inflow series are often of insufficient length to capture severe conditions that may be encountered in the basin. A time series of flows with 30 or 40 years on the record most likely does not include a 100 year wet or dry hydrologic event. Sometimes there is a need to extend the historic series, which is usually done either by (i) the use of regional analysis; (ii) development of rainfall-runoff models, since rainfall data are typically of longer record length than the hydrometric data; and, (iii) the use of stochastic models to generate stochastic inflow time series which are statistically similar to the historic records. The stochastic component relates to the variation of inflows and water demands. This variation is included in the long time series of data (either historic or stochastic), hence the term *implicit stochastic models*. The use of deterministic optimization models over long time series of inputs is typically referred to as *implicit stochastic optimization* (Ilich and Simonovic, 2000a).

The first modelling attempts were combined optimization and simulation models (Bhaskar and Whitlach 1980). Various mathematical programming techniques have been used to address optimal reservoir operation. Russell and Campbell (1996), Bijaya et al. (1996) experimented with fuzzy programming, while Oliviera and Loucks (1997) used genetic programming to optimize reservoir releases in a multi-reservoir system. Perhaps the most popular approach to date is the one based on the use of Linear Programming network solvers such as the Out-of-Kilter algorithm (Ford and Fulkerson, 1962; Barr et al. 1974) or EMNET (Brown and McBride, 1984), which were the basis for several popular models already mentioned in Chapter 3.

The problem in this study exhibits complex non-linearity in constraints, which rules out the use of linear programming. The use of SFEP algorithm is investigated in this study in the context of solving network flow distribution over multiple time steps, where reservoir releases are associated with decision variables. The problem definition is provided first, followed by methodology and a discussion related to the final results.

8.2 Problem Definition

The goal is to optimize operation of a small system consisting of the Bighorn and Brazeau reservoirs (Ilich and Simonovic, 2000b) and their respective hydro power plants. These are the main peaking power generation plants for Transalta Utilites Corporation (TA) that produce a large portion of the peak power requirement in the Province of Alberta. They are located in the North Saskatchewan river basin. The schematic representation of the system is shown in Figure 8.1.

There are two outlets from Brazeau reservoir. One is a set of two gravitational venturi tubes which are used to provide outflows from the upper part of Brazeau storage. When elevation drops below 959.8 m, the venturi outlets can no longer be used and water must be pumped out of the reservoir to Brazeau canal.

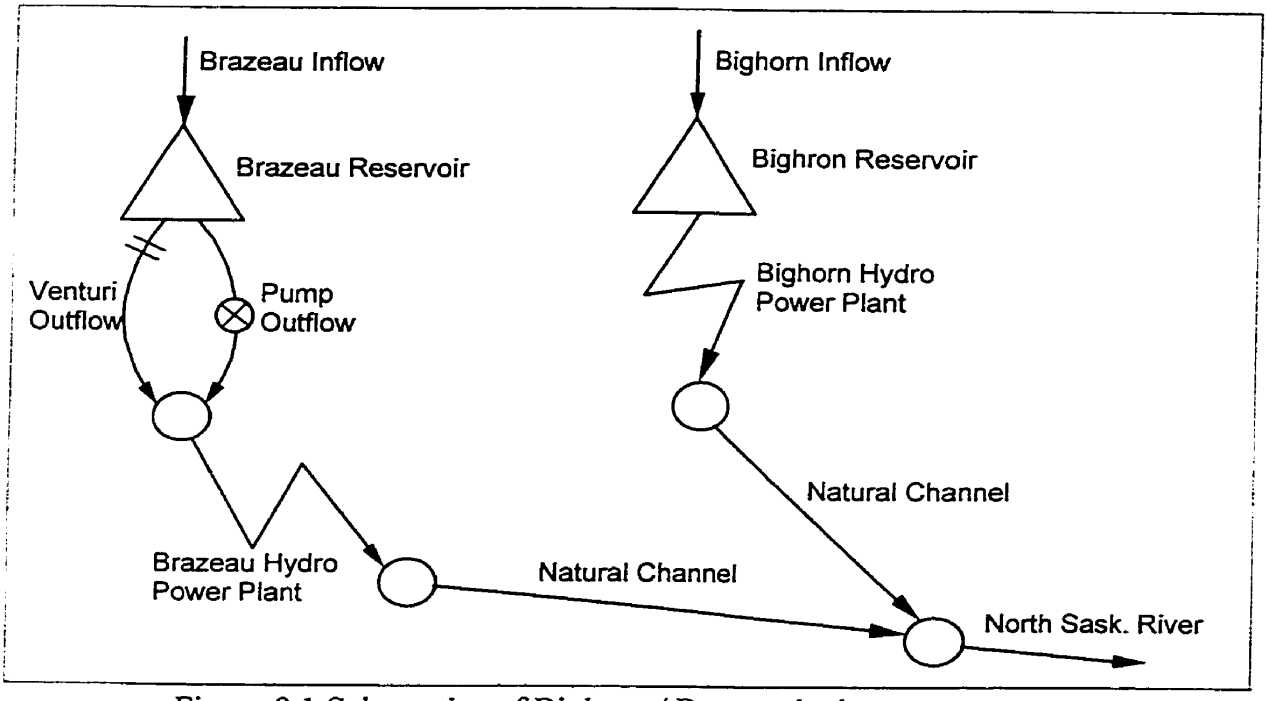


Figure 8.1 Schematics of Bighorn / Brazeau hydro power system

Two pumps of similar capacity are used for that purpose. Both the venturis and pumps release water from Brazeau reservoir into a long canal. The canal is operated in a narrow elevation range of 959.35 m to 959.5 m, thus providing a relatively constant head for the power plant located at its downstream end. The penstock starting at the end of this canal supplies flows to Brazeau hydro power plant. The canal storage change effect can be taken into consideration in hourly or daily operation, but it becomes insignificant in weekly or bi-weekly operation as canal inflows and outflows exhibit a close match. Therefore, one can assume that the power generation at Brazeau is a function of only the flow through the plant, and not head since the head has a relatively constant value. This assumption can be verified by examining the relationship between the historic flow and power generation at Brazeau plant which demonstrates a strong correlation (R^2=0.997). Maximum flow through the Brazeau plant is limited to 347 m^3/s.

The outlet from the Bighorn reservoir flows directly into the Bighorn hydro power plant. The generated power is a function of both flow and the head, where the head can vary from

52 m to 92 m. The maximum design flow through Bighorn plant is 160 m^3/s.

In an effort to increase productivity, TA has initiated the development of new operating guidelines to minimize the cost of pumping. There are two facets related to pumping costs: demand charges and energy charges. Of those, demand charges are more significant. The demand charge is the maximum instantaneous power calculated as the greater of the constant demand or the maximum demand that occurred in the last 12 months. It is obvious that TA should make every effort never to use both pumps simultaneously, in order to reduce the demand charges. This assumption has been incorporated in this study.

The study relied on the 14 years of power generation and hydrologic inflow data in the 1985 to 1999 period. More inflow data are available, however the historic power production patterns were readily available in electronic format only since 1985. Also, the use of power generation data prior to 1985 is less attractive as the power generation patterns of more than 15 years ago would poorly represent the current conditions.

This study was based on using a weekly time step. The basic concept in this study was to allocate power generation between the two plants such that: a) the historic power production pattern is matched as close as possible; b) only one pump is used for pumping water out of Brazeau storage at any point in time; and, c) pump use at Brazeau is minimized. The above goals were to be achieved assuming only one week of perfect forecast of reservoir inflows, the initial reservoir levels at the beginning of the week and the total power production target for the week.

The goal is to minimize the total pump usage. Assume that each unit (1 m^3/s) of flow through the pump is associated with a cost of P, where P is a constant that can take on any finite value, e.g. P = 100. Denoting with Qp[i] the flow through a pump in week i, the above objective can be mathematically formulated as follows:

$$\min \sum_{i=1}^{52} Q_p(i) \cdot P \tag{8.1}$$

The objective function has a straight forward linear formulation, which can also be re-written with constant P taken out of the summation. Constant P represents the cost of pumping 1 m^3/s of flow from Brazeau reservoir into Brazeau canal. Virtually any non-negative value of P can be used. The above formulation demands minimum use of the pump in the operation over the whole year (i.e. 52 weeks, where i is the week counter). There are eighteen constraints in this problem. They are mutually interlinked with non-linear relationships, as listed in expressions (8.2) through (8.19) below.

$$G_{bz}(i)+G_{bg}(i) = P_r(i) \tag{8.2}$$

$$G_{bz}(i) = 23.6877 \cdot [Q_p(i)+Q_v(i)] - 42.66223 \tag{8.3}$$

$$G_{bz}(i) \geq 28.4 \tag{8.4}$$

$$G_{bg}(i) = 0.19344 \cdot Q_{bg}(i) \cdot H_{bg}(i) - 50.16326 \tag{8.5}$$

$$Q_{bg}(i) - I_{bg}(i) = [V_{bg}(i-1) - V_{bg}(i)] \ \frac{1}{T} \tag{8.6}$$

$$V_{bg}(i) \leq V_{bgmax}(i) \tag{8.7}$$

$$H_{bg}(i) = [E_{bg}(i-1)+E_{bg}(i)] \frac{1}{2} \ - T_{bg}(i) \tag{8.8}$$

$$E_{bg} = f_1(V_{bg}) \tag{8.9}$$

$$T_{bg} = f_2(Q_{bg}) \tag{8.10}$$

$$Q_p(i)+Q_v(i) - I_{br}(i) = [V_{br}(i-1) - V_{br}(i)] \ \frac{1}{T} \tag{8.11}$$

$$V_{br}(i) \leq V_{brmax}(i) \tag{8.12}$$

$$0 \leq Q_p(i)+Q_v(i) \leq 347 \tag{8.13}$$

$$Q_v \leq f_3(E_{br}) \tag{8.14}$$

$$Q_p \leq f_4(E_{br}) \tag{8.15}$$

$$E_{br} = f_5(V_{br}) \tag{8.16}$$

$$0 \leq Q_{bg}(i) \leq 160.4 \tag{8.17}$$

$$G_{bg}(i) \geq 0 \tag{8.18}$$

$$Q_{bg}(i)+Q_p(i)+Q_v(i) \geq 14.16 \tag{8.19}$$

where:

$G_{bz}(i)$	average daily energy generated at Brazeau power plant in week i (MWh/day)
$G_{br}(i)$	average daily energy generated at Bighorn power plant in week i (MWh/day)
$P_r(i)$	average daily energy requirement for the two plants in week i (MWh/day)
$Q_p(i)$	pumped flow out of Brazeau lake in week i (m^3/s)
$Q_v(i)$	venturi flow out of Brazeau lake in week i (m^3/s)
$Q_{bg}(i)$	turbine flow through Bighorn hydro power plant in week i (m^3/s)
$I_{bg}(i)$	inflow into Bighorn reservoir in week i (m^3/s)
$H_{bg}(i)$	net head available to the Bighorn power plant (m) in week i, as defined in expression (8).
$V_{bg}(i)$	volume in Bighorn reservoir at the end of time step i (m^3)
$V_{bgmax}(i)$	maximum volume in Bighorn reservoir for the end of each week i (m^3)
T	length of the weekly time step in seconds (7·86400=604800)
$E_{bg}(i)$	ending elevation of Bighorn reservoir for week i
$T_{bg}(i)$	average tail water elevation at Bighorn plant for week i
f_1	a mapping function that converts volume into elevation for Bighorn reservoir
f_2	a mapping function that converts turbine flow into tail water elevation at the Bighorn hydro power plant outlet
$I_{br}(i)$	inflow into Brazeau reservoir in week i (m^3/s)
$V_{br}(i)$	volume in Braeau reservoir at the end of week i (m^3)

$V_{brmax}(i)$	maximum volume in Brazeau reservoir for the end of week i (m^3)
f_3	a mapping function that converts average elevation in Brazeau reservoir over week i into maximum venturi outflow for week i
f_4	a mapping function that converts average elevation in Brazeau reservoir over week i into maximum pump outflow for week i
f_5	a mapping function that converts volume in Brazeau reservoir into elevation

Each of the expressions (8.2) through (8.19) representing a constraint is explained in the following.

(8.1) Objective function formulation.

(8.2) Total power produced at both plants must equal the specified target for each week

(8.3) An empirical relationship between total flow through the turbines (equal to the sum of pumped flow and venturi flow) and the power generated at Brazeau. This relationship is derived using the average weekly historic values and it implicitly includes the efficiency.

(8.4) Minimum average weekly power generated at Brazeau must be 28.4 MW. This can alternatively be specified as the minimum turbine flow through Brazeau being no less than 3 m^3/s (fish habitat requirement).

(8.5) Bighorn power production equation which converts the available net head and the average turbine flow into power. This is an empirical relationship based on the historic data and it implicitly includes efficiency.

(8.6) Water balance equation for Bighorn reservoir for a 7 day time step (inflows - outflows = storage change). All terms in the equation are in the flow units.

(8.7) Limitations on maximum volume in Bighorn reservoir as a function of time. This is dynamic since it is necessary to conduct the pre-flood draw down and refill in the spring.

(8.8) Calculation of net head for Bighorn hydro power plant, which equals average elevation of Bighorn reservoir over a time step minus the average tail water elevation

over the same time step.

(8.9) A function mapping volume of water stored in Bighorn reservoir to surface water elevation.

(8.10) A function mapping turbine flow at Bighorn hydro power plant to surface water elevation below the plant.

(8.11) Water balance equation for Brazeau reservoir for a 7 day time step.

(8.12) Limitation on maximum volume in Brazeau reservoir as a function of week i.

(8.13) Maximum turbine flow at Brazeau must be non-negative and less than 347 m^3/s.

(8.14) Maximum flow in venturis cannot exceed the values dependent on elevation in Brazeau determined by function f_3 which converts elevation to the maximum venturi outflow. The elevation entry into this mapping function should be the average elevation per week, which is equal to $[E_{br}(i-1)+E_{br}(i)]/2$, where $E_{br}(i)$ is the ending elevation per week i.

(8.15) Maximum pump flow cannot exceed the values dependent on elevation in Brazeau, The elevation entry into this mapping function should also be the average elevation per week, which is equal to $[E_{br}(i-1)+E_{br}(i)]/2$.

(8.16) A function for mapping volume of water stored in Brazeau reservoir to surface water elevation.

(8.17) Maximum non-negative turbine flow at Bighorn power plant is limited to 160.4 m^3/s

(8.18) Bighorn generation (MWh/day) must have a non-negative value.

(8.19) Combined release must give a minimum downstream flow of 14 m^3/s.

The unknowns are $Q_{bg}(i)$, $Q_v(i)$ and $Q_p(i)$ for all 52 weeks ($i = 1,52$). The weekly inflow series as well as the historic power generation for both reservoirs were used in the initial attempt to solve the above mathematical program. The output from this exercise is a set of turbine flows at both plants for all 52 weeks which minimize the use of pumping while in the same time meet constraints (8.2) through (8.19).

8.3 Methodology

8.3.1 Optimization of Historic Reservoir Operations

Mathematical program defined by expressions (8.1) through (8.19) is initially solved with a perfect foreknowledge of inflows and power requirements for all 52 weeks in each of the 14 years of record. The goal of this exercise is to use solutions obtained in such a way as a basis for developing operating rules with short term forecasts of inflows and power requirements. Historically, the operation of the system was carried out with occasional use of two pumps simultaneously. There is no guarantee that the historic power production can always be repeated in the model, which relies on the use of only one pump at any point in time. Hence, condition (8.2) was replaced by

$$G_{bz}(i) + G_{bg}(i) \leq P_r(i) \tag{8.20}$$

while the objective function received one more term:

$$\min \sum_{i=1}^{i=52} \{Q_p(i) \cdot P + [P_r(i) - G_{bz}(i) - G_{bg}(i)] \cdot 10000\} \tag{8.21}$$

where 10000 represents the high penalty for not achieving the same level of generation for each week during the historic period under investigation. With this term the objective function also becomes non-linear, since the decision variables are flows, and $G_{bg}(i)$ is a non-linear function of flow.

The problem was solved using a network representation of the system and the existing non-linear relationships in the constraints and the objective function. The SFEP non-linear network solver allows simultaneous non-linearities in the constraints and in the objective function.

To appreciate the complexity of the problem, consider for example a set of perfect solutions

for Bighorn turbine flows $Q_{bg}(i)$. The model is asked to find 52 values, where each of them can range from 0 to 160.4 m^3/s, such that conditions (8.2) through (8.19) are also satisfied (note that these conditions include reservoir routing for both venturi and pump outlet to find the maximum outflow from Brazeau). One small deviation from the best value in week 3 can limit the choices in the remaining 49 weeks. There is no known search method that can guarantee to find the global optimum across the entire time domain, and the SFEP algorithm does not make this claim either. It should also be noted that TA did not use any optimization models to improve its operation in the 1985 to 1999 period.

The initial runs resulted in fourteen annual solutions for the entire system with perfect 52 week forecast for each simulated year. To give more options to the optimizer, each year is started at the beginning of week 26, which corresponds to July 1 for leap years or July 2 otherwise. At the beginning of July the reservoir levels are high and the model has a wider range of operational options to examine as opposed to starting the simulated year on January 1, when the reservoir levels can be low and the range of options is severely restricted. The average potential savings in pumping energy (MWh) achieved with the initial simulation runs were in the order of 60% in comparison to the historic operation. The summary of the results is depicted in column B of Table 8.1. Column A gives the historical pump use in the 1985 to 1999 period.

The final convergence in most cases resulted in the Bighorn elevation being too low, as well as Bighorn reservoir being empty for several consecutive weeks, which would normally be avoided. Rather than rank the best solution solely based on the value of the objective function, a near optimal solution was selected using both the value of the objective function and the shape of the resulting draw down and refill curves. Two judgmental criteria were of interest when analysing the reservoir elevation patterns. First, the ending elevation for the Bighorn and Brazeau should be equal to or higher than the minimum elevation of the historic 1985-1999 record on July 1 for either reservoir. Second, neither reservoir should be empty for longer than one week. Taking all this into considerations, the selected results should be

considered as "near optimal". The results of the above simulated operation with the SFEP optimizer driving power generation at both plants were subjected to two types of analysis, as explained in the following.

8.3.2 Development of Reservoir Operating Zones

Since 14 years is not a large enough sample for statistical analysis, the simulations were repeated for each year individually by selecting a different starting elevation at the beginning of the year for each reservoir. The only exception to this was 1998/99 when the initial levels on July 1 were very high, generation was also high, and incoming flows were below average. If the starting levels in July 1 of 1998 were reduced by 2 metres or more on each reservoir, the model would have serious difficulties meeting the historic production pattern. This added set of simulation runs gave 40 simulated years (3 starting levels for 13 years and one for 1998), which is a slightly better base for statistical analysis of simulation results.

It is possible to look at each individual time slot and find the 10, 20, 50, 80 and 90 percentile elevations from obtained results in every week. If all points with the same percentile are then joined together, they define a draw down and refill line with a certain probability of being exceeded. For example, by joining together the points with 50% probability for each week, a 50% percentile draw down and refill line is obtained. This line tells us that it is the most likely elevation for every point in time during the year for a given reservoir, since 50% of simulated elevations were above it and 50% were below. Operating zones shown in Figures 8.1 and 8.2 were defined for 10, 33, 67 and 90 percentile using this approach.

The operating rules are simple to follow. If at any point in time the operator is considering drawing elevation down from zone 1 in Bighorn reservoir into zone 2 due to high power requirement and low inflow, he must first make sure that Brazeau reservoir elevation is also brought to the bottom of zone 1 before any releases from zone 2 are made at Bighorn reservoir. This rule extends for any other two adjacent zones. Therefore, the policy is as

follows: empty zone 1 at Bighorn storage first, followed by zone 1 at Brazeau, zone 2 and Bighorn, zone 2 at Brazeau, zone 3 at Bighorn, zone 3 at Brazeau, zone 4 at Bighorn and finally zone 4 at Brazeau. The refill rules are very much the same but the order of priority is reversed (i.e. zone 4 at Brazeau is re-filled first, followed by zone 4 at Bighorn, zone 3 at Brazeau, etc.)

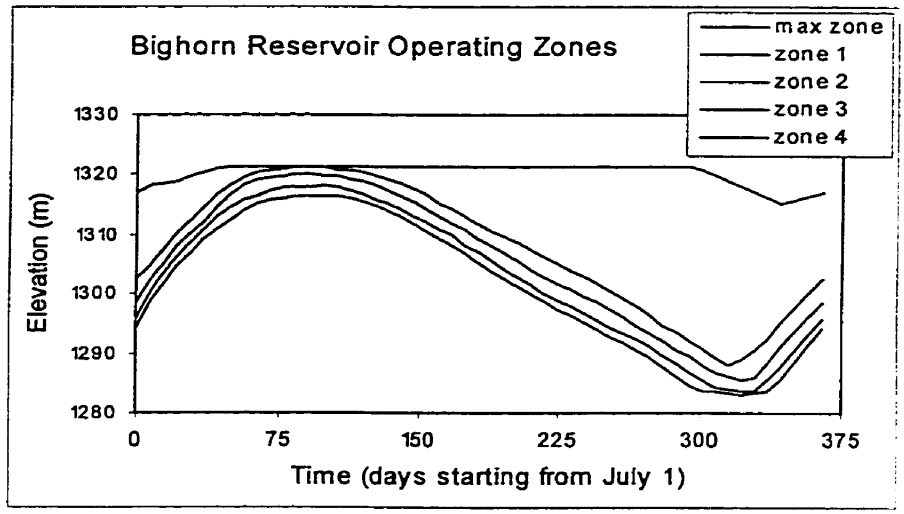


Figure 8.1 Bighorn Reservoir Operating Zones

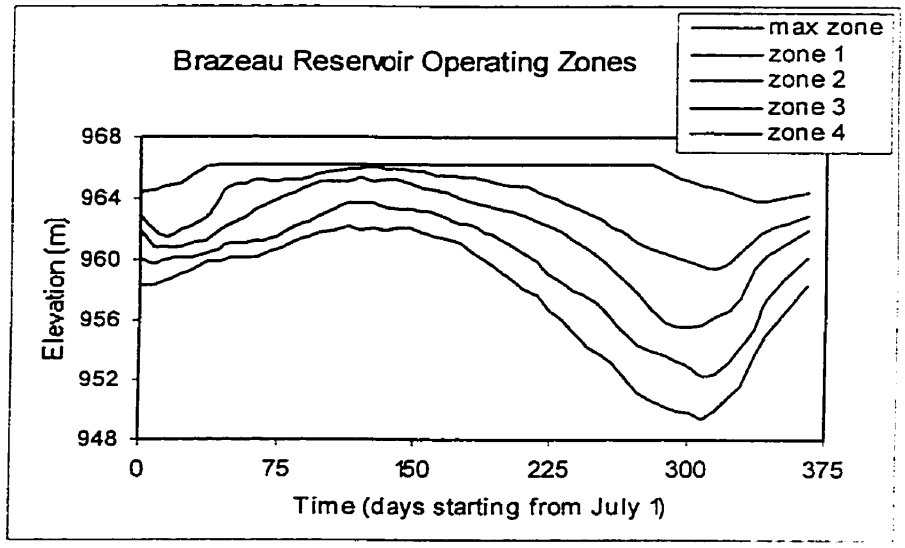


Figure 8.2 Brazeau Reservoir Operating Zones

These zones can be used in short term simulation with only one week forecast. The number of zones is arbitrary, in this case five zones were used on each reservoir. The model was re-run using two significant changes:

1. Only one week forecast was implemented. That means that the model has the inflow and power requirements known for only one week ahead.
2) The objective function was modified to include the reservoir zones, as listed below:

$$\min \sum_{i=1}^{i=52} \{Q_p(i) \cdot P + [P_r(i) - G_{bz}(i) - G_{bg}(i)] \cdot 10000 + C_{br}(V_{br}(i)) + C_{bg}(V_{bg}(i))\} \qquad (8.22)$$

where functions C_{br} and C_{bg} are well known piece wise linear functions representing the value of reservoir storage already discussed at length in the literature (Sigvaldason, 1976).

The results of the simulation run that rely on the above zoning rules while using only one week perfect forecast of inflows and power requirements are encouraging, as shown in column C in Table 8.1. The pump energy savings are close to 50% , which is not as good as it was with a perfect annual forecast, but it is still considerable, especially since this method can easily be applied to systems with many reservoirs and hydro power plants. However, better results can yet be achieved in this case due to having only two reservoirs, as explained below.

8.3.3 Regression Analysis

Since there are only two reservoirs and two hydro power plants, it would make sense to look for a relationship between the turbine flows at each of them as a function of the starting reservoir levels, inflows and the total power requirement.

The goal is to develop a mathematical form of function **f** which links the turbine flows at each reservoir to the inflows, starting reservoir levels and total power requirement for any

given week. This function is based on statistical analysis of the results obtained from 52-week simulation for each year with various starting levels.

$$Q_{bg} = f_{bg}\{E_{bg}, E_{br}, I_{bg}, I_{bz}, P_r\} \tag{8.23}$$

$$Q_{bz} = f_{bz}\{E_{bg}, E_{br}, I_{bg}, I_{bz}, P_r\} \tag{8.24}$$

Assuming reasonably accurate forecasts of inflows and the total load requirement over a week, functions $\mathbf{f}_{bg}$ and $\mathbf{f}_{bz}$ could be used to tell the operator how to best combine releases to produce the required power. In fact, due to constraint (8.2), only one of the functions is required as the other reservoir release can be obtained to match constraint (8.2).

A multitude of values on both the right and left hand side of equations (8.23) and (8.24) are available from a total of 40 simulated years. It was found that the best multiple regression ($R^2 > 0.995$) can be found between the Bighorn reservoir level at the end of a week as a function of the initial elevations at Bighorn and Brazeau, average weekly inflows into both reservoirs and the total power requirement for a week. The relationship has the following form:

$$E_{bg}(i) = a\cdot E_{bg}(i-1)+b\cdot E_{bz}(i-1)+c\cdot P_r(i)+d\cdot I_{bg}(i)+e\cdot I_{bz}(i)+f \tag{8.25}$$

where symbol E represents the ending elevation for a week while index i represents the week counter. Hence the ending elevation for week i-1 is the starting elevation for week i. Coefficients a, b, c, d, e, and f have been determined as 0.994422, 0.0564, −0.000472, 0.0165, 0.0073 and −47.55229, respectively.

The predictive regression model gives the week ending elevation for Bighorn reservoir given the initial elevations and weekly inflows for both reservoirs, along with the total power requirement for both hydro power plants. In combination with Bighorn inflow and the initial elevation, the predicted reservoir level from the multiple regression can be used to estimate reservoir outflow for a week. The multiple regression model was tried instead of the

reservoir operating zones model discussed above, however the trial failed to produce good results. In some cases regression was causing reservoirs to spill, and in a few rare cases it resulted in negative turbine flows at Bighorn. It was decided that the relationship developed using multiple regression cannot be used on its own.

The final avenue that was attempted was to combine multiple regression relationship with the zoning concept developed in the previous step. This is a promising approach for a number of reasons:

a) The search for good initial solutions for each week can be improved using the regression relationship. It is not necessary to generate any values of Bighorn outflow between 0 and 160.4, but rather focus on a much smaller search space defined in each step dynamically using the multiple regression defined by equation (25). The regression gives the most likely guess, and the other guesses are clustered around it using normal distribution with a small standard deviation equal to 10% of the value of the initial guess. The importance of generating initial parent solutions in favourable areas of the feasible region has been emphasized earlier in Chapter 7. In this case the favourable region is not determined using the shape of the objective function, but with the multiple regression instead.

b) Unintelligible guesses (the ones that result in spills or negative outflows) are discarded using the zoning concept which forces generated trial solutions to conform to the stated zoning use policy, as well as the recombination mechanism which throws away solutions with poor fitness.

8.4 Results

The final solutions in column D of Table 1 obtained from a combinatorial model consisting of the above zoning concept in combination with the multiple regression predictive model gave excellent results with only one week forecast of inflows and power requirements. The

average reduction of energy required for pumping is around 88% in comparison to the actual historic record. In terms of the actual cost savings in dollars, it could be even higher than 88% since a lower demand charge could be negotiated due to the assumed operation of only one pump instead of two.

The results listed in column D of Table 8.1 surpassed the performance of the model displayed in the initial runs with 52 week foresight. This means that the initialization procedure (the first step in the search process) is very significant for the success of the model. In this case there are two issues that preclude finding the best solutions using the 52 week foresight:

1) The use of high penalty for not meeting the total power demand from both plants (conditions 8.20 and 8.21) is fairly frequent in the first set of simulations with 52 week forecast, however it declines significantly once the operating zones are introduced. This allows the model to focus more intently on the search within the true feasible region without wasting time with infeasible solutions that had to be filtered out using a large penalty. This is yet another argument against the use of penalty functions in this kind of search.

2) Initial selection of solutions with 52 week forecast was not only based on the value of the objective function, but also on the length of period Bighorn reservoir was empty. In most cases this meant sacrificing optimality by about 25 % and taking solutions which have not yet converged such that the dam operators can accept the "role model solutions" as building blocks for generating short term operating policy. Therefore, solutions in column B above are a mix of art and science, they are not the best solutions found, but they are the best solutions that would likely be acceptable to the operators.

This above conclusions also mean that other search options should be examined in the future

to achieve higher optimality in the initial step of this process, especially when penalty functions are required to ensure convergence to feasibility of the final solutions.

Table 8.1 Summary of historic and simulated pumping energy requirements in MWh

Column	A	B	C	D
Year	Historic pumping	52-week forecast pumping	one week forecast & zones	zones & regression rule
1985/86	20472	17540	16807	2483
1986/87	18315	1620	2183	204
1987/88	13825	5623	9223	2577
1988/89	14783	3422	5873	393
1989/90	23255	1711	5036	5
1990/91	23094	6364	9974	2294
1991/92	20301	13878	15983	3289
1992/93	16512	8633	9234	1011
1993/94	21515	14835	15831	1514
1994/95	17501	11979	14086	1910
1995/96	15511	4155	7292	886
1996/97	15227	1860	5081	1273
1997/98	13154	1630	4860	3103
1998/99	18540	11435	12815	9043
Average	18000	7482	9591	2142

The problem has three variables – venturi and pump flow at Brazeau along with the turbine flow at Bighron power plant. These three variables must initially be solved for all 52 time steps simultaneously, so there are 156 variables along with 19 constraints for each time interval, which gives a total of 19 x 52 = 988 constraints, virtually all of which are non-linear. The SFEP solver was capable of solving the above problem within 13 seconds on a 500 MHz PC running the Windows 95 operating system. The initialization procedure included a total of 1000 individuals. The mating pool has 10 parents, and the recombination was carried out for up to 3000 trials. Table 8.2 gives the objective functions of the first 10

individual solutions, and the best 10 solutions left in the mating pool after 1000 randomly generated solutions. This is followed by the makeup of the mating pool after each 500 new solutions were generated using recombination. Solution to this problem could not be obtained using other solvers. Microsoft Excel could not handle more than 200 non-linear constraints, while the MATLAB package failed to deliver an intelligible solution even though the technical support staff tried to help in setting it up for this problem (Ilich, personal communication).

Table 8.2 Values of the objective function during the SFEP progression

	INITIALIZATION		RECOMBINATION				
solution rank	initial 10 solutions	additional 990 solutions	500 solutions	1000 solutions	1500 solutions	2000 solutions	2500 solutions
1	19016.679	9756.920	6397.354	5972.465	5872.732	5744.552	5716.279
2	40319.663	9767.353	6427.565	6042.506	5874.132	5769.507	5731.533
3	48629.504	9847.628	6427.629	6051.479	5890.697	5804.777	5744.364
4	50986.731	10052.935	6434.765	6062.557	5891.494	5818.105	5744.552
5	53156.772	10052.935	6463.150	6073.715	5896.003	5821.235	5752.498
6	60609.446	10064.275	6474.816	6080.030	5903.187	5821.643	5755.371
7	62221.291	10141.037	6475.504	6084.495	5905.762	5826.763	5756.664
8	63660.235	10212.207	6477.797	6085.910	5911.159	5835.306	5756.738
9	75397.845	10368.341	6482.027	6103.362	5916.579	5835.579	5757.006
10	119142.199	10543.934	6482.272	6110.854	5919.726	5835.969	5757.383

There are no units in the above formulation, since the penalty for pumping a unit of flow in expression (8.1) is user-defined, while violating condition (8.2) even by a small amount incurs a large penalty of 10000, which is also set arbitrarily.

8.5 Conclusion

The SFEP algorithm is used in this study in an effort to minimize pumping from lower storage of Brazeau reservoir. The promising feature of this approach is the inclusion of non-linear functions related to reservoir routing and hydro power generation directly in the search process. The generated rule based on a one week forecast of inflows and energy requirements and the use of the SFEP solver to simulate operation for each week individually show potential pump energy savings of close to 90% on average over the last 14 years. This

approach can be used on other systems with different objectives, which can include maximizing energy generation in combination with other in-stream or off-stream water requirements.

9 CASE STUDY III – WATER ALLOCATION IN THE BRANTAS RIVER BASIN IN EAST JAVA, INDONESIA

9.1 Introduction

This Chapter describes testing of the SFEP solver on the Brantas river basin depicted in Figure 9.1 and located in East Java, Indonesia. The goal was to use the SFEP such that the model could find water allocation that best meets the established objectives of Perum Jasa Tirta (PJT), a water management agency in charge of regulating the major water intake structures and reservoirs in the Brantas river basin.

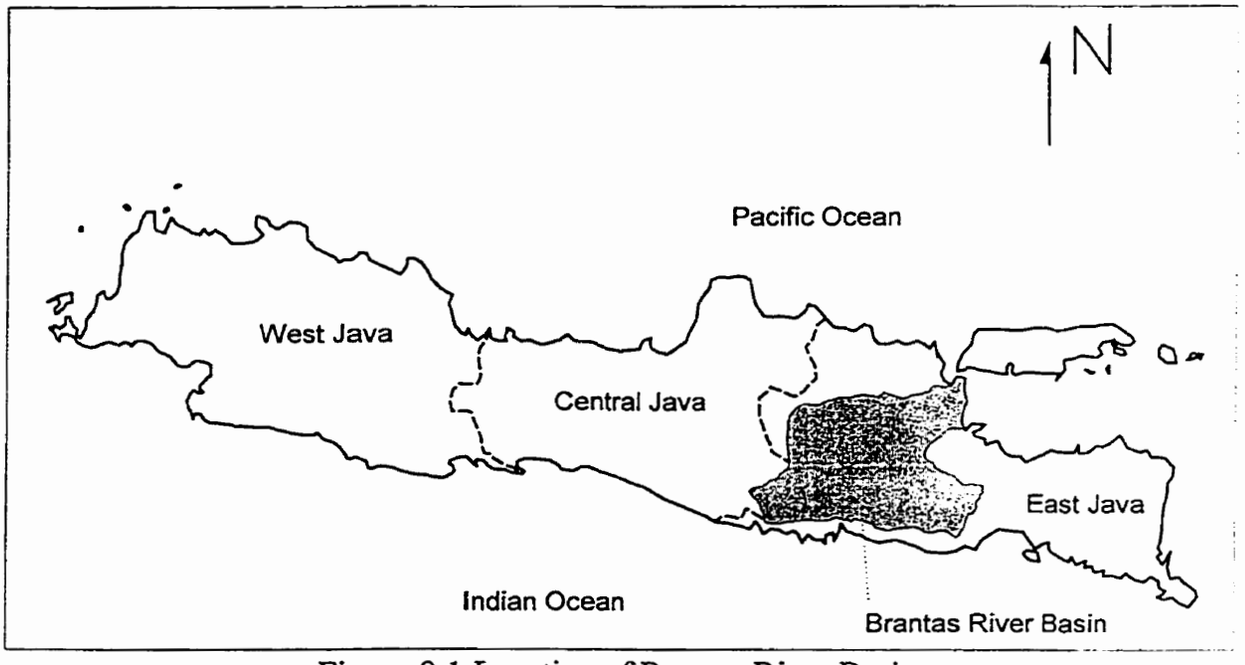


Figure 9.1 Location of Brantas River Basin

Perum Jasa Tirta (PJT) has long been a role model agency regarding water resources management in Indonesia. The agency is financed mainly by charging fee for water use in the basin. There are four types of users in the basin: hydro power producers, municipalities, industries, and agriculture. Although the present situation is in the process of change, the current fee structure is lenient towards farmers, whose irrigation withdrawals account for most of the consumptive water use, while they do not pay for it. Other water users in the

basin are levied commercial rates which is aimed to encourage water use efficiency and sound management practices.

While irrigation, municipal or industrial water use all have upper limits on demand which vary in time and space, there is no such limit explicitly stipulated for hydro power, where the goal is to maximize power generation. Maximizing total power generation in the basin over the whole year is a desirable objective since PJT collects a percentage of the generated power as its revenue, although it has been noted that there is a surplus of electric energy in Indonesia at present and the goal of maximizing power output from the Brantas basin may necessarily coincide with the overall government objectives. This objective is constrained physically by the flow and head capacities of the turbines, and operationally by other priorities that may take precedence. Hence, the management objective for the Brantas basin (and many other similar river basins) can be mathematically expressed as:

$$\max\left\{\sum PR_p - \sum Q_p C_p + \sum Q_{ir} R_{ir} + \sum Q_{in} R_{in} + \sum Q_m R_m + \right.$$
$$\left. + \sum \left(Q_{rm} - D_{rm}\right) C_{rm} \right\} \qquad (9.1)$$

where:

P	power generation at any of the hydro power plants in the basin (kwh)
Rp	revenue per kwh allocated to PJT as per the existing agreement (Indonesian Rupiah = Can$0.00019 at the time of this study)
$\sum$ P · Rp	sum of revenues from all hydro power plants in the basin (Rp)
Qp	pumped flow through a pumping station (m^3/s)
Cp	cost of pumping per 1 m^3/s assuming constant head rise (Rp)
$\sum$ Qp · Cp	sum of all pumping costs in the basin (Rp)
Qir	water supply for irrigation (m^3/s)
Rir	added value of crop production due to 1 m^3/s of irrigation supply (Rp)
$\sum$ Qir · Cir	added value of total crop production from all irrigated areas (Rp)

Qin	water supply for industry (m^3/s)
Rin	industrial fee per 1 m3/s of water use (Rp)
$\sum$Qin · Cin	total revenue from all industrial water use in the system (Rp)
Qm	water supply for municipalities (m^3/s)
Rm	municipal fee per 1 m3/s of water use (Rp)
$\sum$Qm · Cm	total revenue from all municipal water use in the system (Rp)
Qrm	water supply for riparian needs (river maintenance) (m^3/s)
Drm	river maintenance flow target (m^3/s)
Crm	the cost of damage caused by not meeting the riparian flow requirement by having 1 m3/s deficit (Rp)
$\sum$[(Qrm – Drm) · Crm]	the value of total loss for not meeting the riparian flow targets (Rp)

The fees that are currently is use are 51Rp/m^3, 35 Rp/m^3 and 13.61 Rp/KWh for industrial, municipal and hydro power use, respectively. However, there are legal and operational requirements that may override the economic values attached to them. For example, industrial users pay a higher fee than municipalities. However, municipal supply takes legal precedence over industrial water use, and as such it should be assigned a higher pricing vector indicating higher priority within the model. There are other water management objectives that are political, such as irrigation or the maintenance flow. The fee levied for irrigation water use is still a political issue which brings some uncertainty to the value of Cir. If zero is used as the value for Cir (since irrigators are currently not paying for water at all), water may not be allocated to irrigation within the model. Hence the value of Cir must be determined using its political importance. Similar remark is applicable to Crm, which is the equivalent monetary value associated with river maintenance (this implicitly includes water quality as the maintenance flows may be governed by water quality requirements).

The above objective can be formulated as maximization of revenue for PJT with inclusion of important political objectives and operational constraints. It is applicable to each operational time interval, which in this study is restricted to a 10-day period due to the

available input data. Since an allocation decision in one time interval has implications on the management options in the following time intervals, it is desirable to carry out basin-wide optimization of allocation both in space (due to spatial variation of supply and demand) and in time (due to seasonal variations in supply and limited reservoir storage). Emptying reservoirs too soon may cause high costs to the downstream users later in the dry season, and vice versa, being too conservative in the beginning of the dry season may limit the final output for the whole year by using only a fraction of the available live storage. Hence, the goal of finding optimal allocation must include the time component by summing up the above expression over all time intervals within a year:

$$\max\Sigma\left\{\Sigma PR_p - \Sigma Q_p C_p + \Sigma Q_{ir} R_{ir} + \Sigma Q_{in} R_{in} + \Sigma Q_m R_m + \right.$$
$$\left. + \Sigma\left(Q_{rm} - D_{rm}\right)C_{rm}\right\} \tag{9.2}$$

where the first summation is over all time intervals within a year, while the summations inside the curly braces are conducted over all basin components of the same type (e.g. irrigation, industrial, or municipal users). The above equation maximizes annual net revenue for PJT. The following constraints apply to the above maximization problem:

$$P \leq Pmax(Q, H) \tag{9.3}$$

Power generation is constrained by the operating characteristics of the turbine and generator, which are functions of flow and average net head over a time interval. The average net head is a function of average reservoir inflow, outflow and the starting elevation for a time interval.

$$Qp \leq Qmax(H) \tag{9.4}$$

Maximum pumping rate is constrained by the operating characteristics of the pump, which is also affected by the average anticipated head rise over a time step.

$$Qir \leq Dir \tag{9.5}$$

Irrigation supply should not exceed the ideal demand Dir defined for each area by the crop requirements and conveyance losses.

$$Qin \leq Din \tag{9.6}$$

Industrial supply should not exceed the ideal demand Din (m^3/s).

$$Qm \leq Dm \tag{9.7}$$

Municipal supply should not exceed the ideal demand Dm (m^3/s).

Finally, one more term should be added to the objective function. It represents importance of storing excess water in the reservoirs. Without this term, the model would be indifferent to spilling surplus flows as opposed to storing them in reservoirs. The pricing vector for storage is the lowest in the system, which means that storage will give in to any other demand. The low pricing vector is still required to make sure that reservoirs re-fill during the wet season. With this term the objective function takes the following form:

$$\begin{aligned} \max \sum \Big\{ & \sum P.R_p - \sum Q_p.C_p + \sum Q_{ir}.R_{ir} + \sum Q_{in}.R_{in} + \sum Q_m.R_m + \\ & + \sum \left(Q_{rm} - D_{rm}\right).C_{rm} + \sum \left(Q_r - D_r\right).C_r \Big\} \end{aligned} \tag{9.8}$$

where:

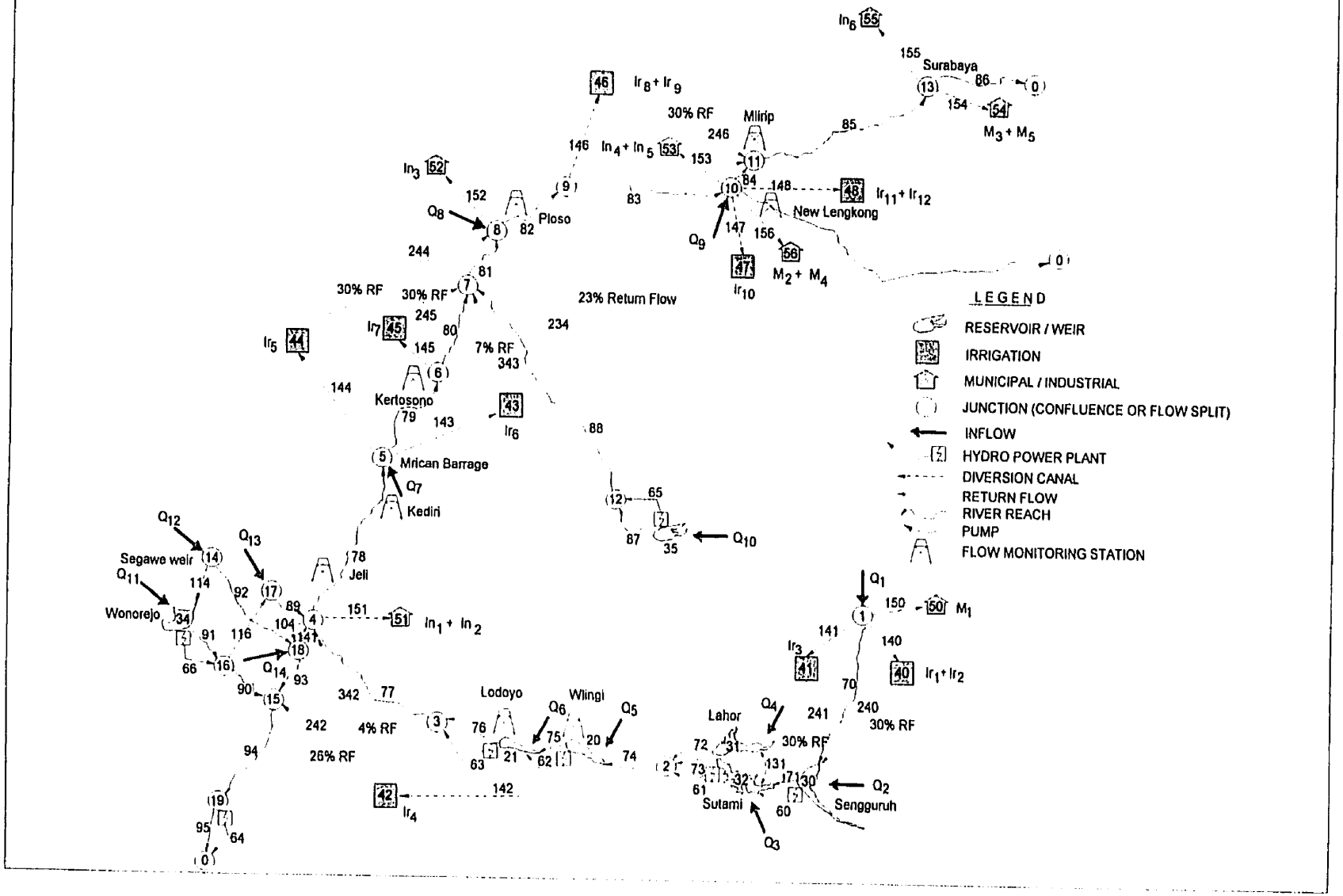
Qr	ending reservoir storage in the units of flow for a time interval (m^3/s)
Dr	target flow into reservoir required to keep it full at the end of a time step (m^3/s)
Cr	the value of storage (i.e. the cost of 1 m^3/s of deficit in storage)
$\sum(Qr - Dr) \cdot Cr$	the value of total cost for deficit in storage for all reservoirs

Since the cost of storage deficit is the lowest in the system, it does not alter the solution (water allocation) for other components. It is required by the model to avoid unnecessary spills and ensure re-fills during high flow seasons.

The model was applied on the current system configuration in an effort to investigate possible management improvements over the last 23 years. Therefore, the sub-system consisting of Wonorejo reservoir numbered (34) in the Schematics in Figure 9.2 has been excluded, along with the entire Wonorejo sub-system, including the pump (141), diversion canals (114), (116) and (104) and hydro power plants (64) and (66). Basically, the Brantas river basin has been cutoff from the Wonorejo sub-system and the new developments presently under way are aimed at re-establishing the connection at node (4). This connection is still not operational so it was left out in this study. Sutami and Lahor are the only two reservoirs with sizeable storage. Sengurruh, Wlingi and Lodoyo are weirs with run-off-the-river hydro power plants. They were modeled as nodes without storage since their entire storage can be re-filled (or emptied) with a flow of less than 3.5 m^3/s over a 10-day time step, which was used in this study. Typical river flow at those weirs in the dry season is around 50 m^3/s while in the wet season it may exceed 200 m^3/s.

Return flow channels return a percentage of gross diversions to the points of return, while inflow nodes represent runoff contribution of a sub-catchment between subsequent inflow nodes. Local inflows can be either positive or negative, implying gains or losses (mainly due to channel attenuation) within the same 10 day period. There are 25 decision variables in the Schematic in Figure 9.2, and they were solved simultaneously for 37 time intervals (one full year plus one more time interval from the following year), such that the actual network size of this problem corresponds to 925 independent variables. In addition to testing the SFEP solver on a problem of this size, the other benefit is to try to examine by how much the Basin operation could have been improved in the past if the minimum riparian flow of 20 m^3/s has been maintained at the city of Surabaya (channel 85 in Figure 9.2).

Figure 9.2 Brantas River Basin Modeling Schematics



Historically, the imposed minimum was only 7 m^3/s. In recent years more water is needed during the low flows seasons due to municipal supply and water quality concerns.

9.2 Modelling of the Brantas Basin with the SFEP

The model was setup to evaluate system performance over individual time steps as well as simultaneous time steps up to one year. This was done to ease the use of the model as an operational tool. PJT is currently using the model as a seasonal planning tool. Of particular interest are the non-linear constraints in this problem, which are reviewed in the following.

9.3 Non-Linear Features

The general form of the objective function was defined earlier in Section 9.1. The term related to pumping can be ignored, since the pump station under construction is a new component that hasn't existed over the last 23 years for which the input data are available. Note that the objective function from Section 9.1 is non linear due to the power term. This is the only non-linear item in the objective function, but it is rather complex since the values of head, flow and efficiency have intricate links among them that should be preserved. Of particular interest are non-linear flow and head constraints which are listed below.

9.3.1 Maximum Turbine Flows and Net Head

Sutami hydro power plant generates about the half of the hydro electric output of the entire Brantas basin. The maximum turbine flows are given as a function of the available average net head (NH) over a time step, where NH is given as:

$$NH = HW_{avg} - TW_{avg} \tag{9.9}$$

HW_{avg} and TW_{avg} are the average head and tail water elevation over a given time step, and they themselves are functions of the initial storage, inflow and outflow from the reservoir for

a given time step. Hence, one could write:

$$HW_{avg} = f\left(\sum Q_{in}, \sum Q_{out}, V_{in}\right) \tag{9.10}$$

$$TW_{avg} = f\left(\sum Q_{out}\right) \tag{9.11}$$

The first expression does not include the ending reservoir storage since this is implicitly included in the three given parameters. The second expression defines outflow as the only factor that derives the tail water elevation through some form of stage-flow rating curve. That relationship is in effect for all hydro power plants in the Brantas basin except in the case of Sengguruh hydro power plant, where TW_{avg} of Sengguruh is equal to the HW_{avg} of Sutami reservoir located downstream of it. In other words, the Sutami lake level defines the tail water elevation for Sengguruh.

The desired choice is to route all outflows ($\sum$Qout) through the turbines. However, this may not be possible since the limit on turbine flows is a function of the available NH, and yet NH depends on the entire reservoir balance ($\sum$Qin , $\sum$Qout and the starting reservoir level for a time step). The maximum flow (QM) function was defined using the polynomial fit based on the turbine data available from manufacturers, resulting in the following:

For NH $\geq$ 78 m

$$QM = 3\ (0.03097220618\ NH^2 - 5.875362193265\ NH + 321.32746923567) \tag{9.12}$$

For NH < 78 m

$$QM = 3\ (0.018590866186\ NH^2 - 2.089819347\ NH + 100.3274692357) \tag{9.13}$$

Multiplier 3 denotes the maximum flow for the case when all three turbines in parallel connection are operated at Sutami hydro power plant. The above relationship has been incorporated into the model for this study.

9.3.2 Hydro Power Efficiency

Power is produced with variable efficiency, which depends on the combination of net head and flow for a given time step. An attempt was made to derive a functional form of efficiency using the historic operation data for Sutami and Selorejo power plants. This attempt failed in the case of Sutami hydro power plant, since the historic data contained errors which often provided efficiencies above 100%. The historic data were used for developing an empirical expression for efficiency as a function of flow and head for Selorejo, while the manufacturer's specifications were used to develop the same kind of expression for Sutami hydro power plant. The expressions are given as follows:

For Sutami hydro power plant:

$$EF = -0.000040699891\,Q^2 + 0.010532122949\,Q - 0.000055418277\,NH^2 + 0.010878934817\,NH - 0.293522823168 \quad (9.14)$$

For Selorejo hydro power plant:

$$EF = 0.004147914\,Q^2 - 0.096314865\,Q - 0.000674151\,NH^2 + 0.055030742\,NH + 0.13167155 \quad (9.15)$$

The above relationships between turbine efficiency (EF), flow (Q) and head (NH) have been included directly in the source code and used each time the power generation was calcualted.

9.3.3 Connection Tunnel

By far the most challenging non-linear constraint is related to the flows in the connection tunnel between Lahor and Sutami reservoirs. These flows typically range between 5 and 10 m^3/s, although they are known to have exceeded 20 m^3/s on rare occasions. The tunnel flows

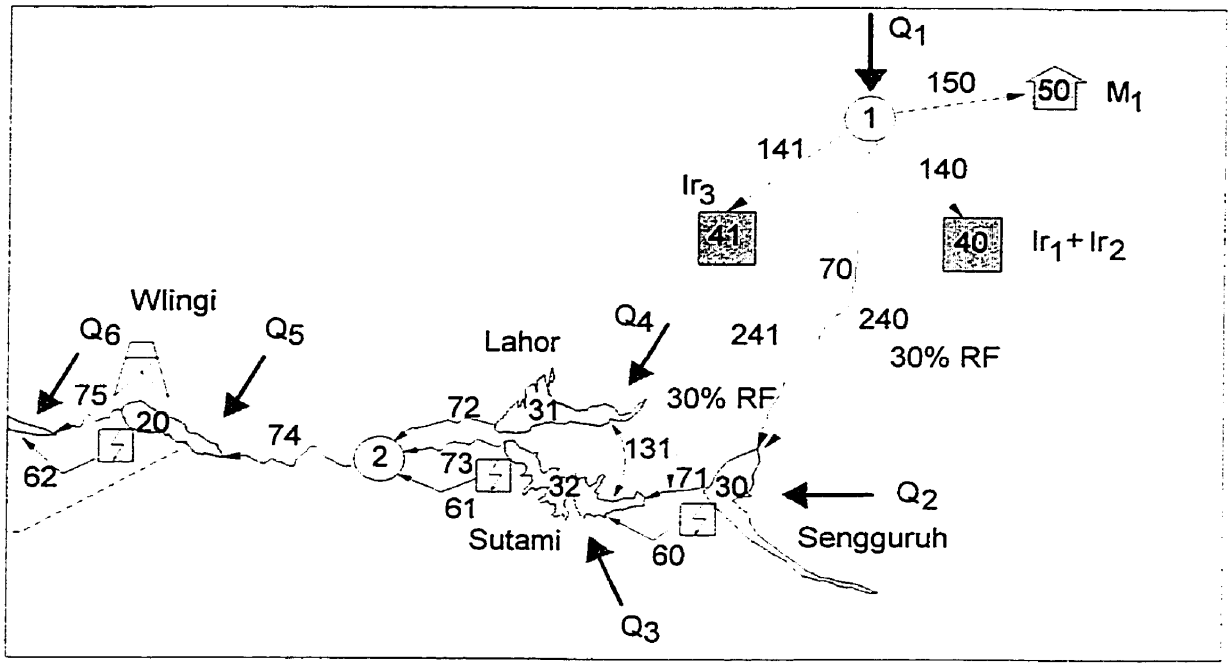


Figure 9.3 Sutami and Lahor connection tunnel (131)

are a constituent element in the water balance equation for both Lahor and Sutami. Proper modeling of those flows is essential for allocation studies and real time operation within the Brantas river basin. Figure 9.3 gives the layout of the two reservoirs. Inflow into Sutami reservoir consists of the return flow from Molek Irrigation (block 41), outflow from Sengguruh weir (power flows through plant 60 along with spills via channel (71) when total outflow exceeds power plant capacity) and the local inflow (Q_3) within the catchment between Sengguruh weir and Sutami reservoir. The only Inflow into Lahor reservoir is the local runoff from its catchment (Q_4). Lahor has two outflows: (a) downstream spill via channel (72), and (b) connection tunnel connection tunnel numbered (131) in Figure 9.2. Flows in the connection tunnel within one time step can be approximated using the following:

$$Q = 10\sqrt{H} \tag{9.16}$$

where H is the average elevation differential between the two reservoirs over a given time interval, i.e.

$$H = \left(\frac{Els + Ele}{2}\right) - \left(\frac{Ess + Ese}{2}\right) \tag{9.17}$$

where:

Els starting elevation at Lahor for a time step (m)

Ele ending elevation at Lahor for a time step (m)

Ess starting elevation at Sutami for a time step (m)

Ese ending elevation at Sutami for a time step (m)

Of those, only Els and Ess are known, while Ele and Ese depend on:

(a) total inflow and outflow for Sutami reservoir;
(b) total inflow and outflow for Lahor reservoir; and,
(c) starting elevation of Lahor and Sutami (Els and Ess).

It should be noted that connection tunnel flows can take any direction between Sutami and Lahor reservoirs. The connection tunnel flow can act as either inflow or outflow in the balance equation of each reservoir. Therefore, the connection tunnel flow is a part of (a) and (b) listed above. The problem of finding the best allocation in the basin considerably complicated by the above functional dependencies, where the connection tunnel flow depends on six other variables listed under (a), (b) and (c), while four of those listed under (a) and (b) are also affected by the connection tunnel flow itself. Note that an error of 0.1 meters in assumed value for the head differential H gives an error in the estimate of the connection tunnel flow in the order of 3.16 m^3/s. The functional dependence is very sensitive to the value of elevation differential H, thus requiring a sophisticated convergence scheme in the process of generating feasible solutions. A feasible solution is the one where the starting and ending levels of both reservoirs give a calculated value of the connection tunnel flow, while the same value of the connection tunnel flow is included in the mass balance equations for each reservoir. Permitted tolerance limit is +/-0.01 m^3/s between the assumed and calculated value of the connection tunnel flow.

9.4 Definition of Modelling Objectives

The objective function that can be used in the SFEP is not limited only to linear terms. Two cases are considered, one referring to a single solution for each time interval and the other referring to a simultaneous solution over the entire year. The following cost factors are considered, expressed in cost for 1 m^3 of deficit from the target for riparian flow and reservoir storage and the revenue from allocating 1 m^3 to industry, municipality, irrigation and finally revenue from generating 1 KWh of hydro power:

Component	Cost Factor (Rp)
riparian flow target	145 / m^3
municipal demand	- 35 / m^3
industrial demand	- 51 / m^3
irrigation demand	- 25 / m^3
reservoir storage	0.06 / m^3
hydro power	- 13.61 / KWh

Negative cost factors indicate benefits (revenue) to PJT, while positive cost factors describe importance (weight) of an undesirable factor. The highest cost is incurred by violating the riparian flow requirement, which is incurred for each m^3 of deficit required to maintain the flow of 20 m^3/s in channel (85). An effort was made to model the existing cost parameters that are currently charged by PJT. In that context, maintenance flow target of 20 m^3/s is considered an important political objective. As such, it has a higher value than industrial, municipal or irrigation water use. According to the current fee structure hydro power has by far the lowest priority. Using Sutami as an example, with rated head of 78 m, unit flow of 1 m^3/s, average efficiency of 0.85 and the cost factor of 13.61 Rp/KWh, the revenue Rs over a 10 day period is:

$$Rs = (9.807 \cdot 78 \cdot 1 \cdot 0.85)\ KW \cdot 10\ days \cdot 24hr/day \cdot 13.61 Rp/\ KWh = 2123826.672\ Rp$$

When expressed in the units of 1000 Rp, this is equivalent to the revenue of 2123 [10^3 Rp] per 1 m^3/s of flow for 10 days. On the other hand, 1 m^3/s of municipal water use would generate revenue of 86.4*10*35 = 30240 in the units of [10^3 Rp]. According to this cost structure, reservoir releases should be made for other water uses and power should be generated as a by-product of those releases. A possible conflict between consumptive use and hydro power is only apparent in the case of the first three irrigation blocks, numbered (40), (41) and (42) in the schematics in Figure 9.2. Other irrigation blocks are located downstream and they could readily utilize flows released from hydro generation throughout the year.

The above fee structure results in a priority policy which is in line with many other river basins where hydro power generation is a by-product of reservoir releases made for other purposes, especially in the dry season.

Case A: Single time step optimization

In this case each time interval is handled individually, which means that the model does not use flow forecast beyond a single 10-day time step. The objective function can be formulated in general as:

$$\max\left\{\sum P.R_p - \sum Q_p.C_p + \sum Q_{ir}.R_{ir} + \sum Q_{in}.R_{in} + \sum Q_m.R_m + \right.$$
$$\left. + \sum\left(Q_{rm} - D_{rm}\right).C_{rm} + \sum\left(Q_r - D_r\right).C_r\right\} \qquad (9.18)$$

Since the SFEP setup on the existing configuration without Wonorejo sub-system, the term related to pumping costs ($\sum Qp \cdot Cp$) can be ignored.

The SFEP was originally setup to solve the minimum cost flow problem in a network with non-linear objective function and constraints. In the above formulation the objective function is set to maximize the net benefit. To convert it to a form suitable for SFEP, the

terms within should change sign and maximization should change to minimization. This is a simple mathematical conversion that has no physical meaning, however mathematically the two forms of the objective function are equivalent.

$$\min\left\{-\Sigma P.R_p - \Sigma Q_{ir}.R_{ir} - \Sigma Q_{in}.R_{in} - \Sigma Q_m.R_m + \Sigma\left(D_{rm} - Q_{rm}\right).C_{rm} + \Sigma\left(D_r - Q_r\right).C_r\right\} \quad (9.19)$$

Cost factors for hydro power, irrigation, industrial and municipal water use are given in the input data file with negative signs which means that the target is to maximize them, while the maintenance flow target and storage conservation have a positive sign since the model is penalizing deficit of deviation from the ideal level for those two components.

Case B: Multiple time step optimization

In this case the objective is the same as in Case A but the entire year is solved at once, which means that the model uses perfect flow forecast for 365 days. Using the same argument as for case A related to the conversion of maximization of net benefit into a minimal cost flow problem, the previous objective function can be expanded over all 36 time intervals simultaneously.

$$\min\Sigma\left\{-\Sigma P.R_p - \Sigma Q_{ir}.R_{ir} - \Sigma Q_{in}.R_{in} - \Sigma Q_m.R_m + + \Sigma\left(D_{rm} - Q_{rm}\right).C_{rm} + \Sigma\left(D_r - Q_r\right).C_r\right\} \quad (9.20)$$

one should keep in mind that there are upper limits to consumptive (irrigation, industrial and municipal) water use. Those three terms could have also been re-written using a positive cost applied to the deviation from the ideal zone. In other words,

$$\min\left\{-\Sigma Q_{in} \cdot R_{in}\right\} \quad (9.21)$$

subject to:

$$Q_{in} \leq D_{in} \tag{9.22}$$

The above formulation is equivalent to

$$\min\left\{\Sigma\left(D_{in} \cdot Q_{in}\right) \cdot R_{in}\right\} \tag{9.23}$$

in the second formulation the constraint is included in the objective function. Mathematically, those two forms state the same objective. At this point the SFEP was programmed to use the former formulation, however that can be easily changed to accommodate either form.

Case B is based on perfect flow forecast for the whole year. This is a major advantage which should give better overall results in Case B than in Case A. The studies involving Case B are of practical use only if they are aimed to develop reservoir operating guidelines.

9.5 Results

The first run was selected on three individual years typical of medium, wet and dry conditions represented by years 1978, 1984 and 1991, respectively. Table 9.1 shows the progression of the SFEP algorithm during the solution process. The first column shows the value of the objective function for a given solution, with corresponding values of only the first four independent variables belonging to that solution. The problem has 925 independent variables (25 for each of the 37 time intervals) and additionally there are 703 dependent variables (19 for each of the 37 time intervals). The entire mating pool therefore consists of 925 columns, too many to show in hard print. However, Table 9.1 is primarily of interest in terms of the change of the values of the objective function. The year that was chosen in Table 9.1 is 1991 (dry year) with simultaneous solution for the whole year. Channels (131), (72) and (73) are the connection tunnel, Lahor reservoir spill and Sutami reservoir spill, respectively, while component 61 is hydro power plant at Sutami reservoir (the values shown

in the table are all in flow units). Inflow into Lahor in the first time step is 7.48 m^3/s (=4.74+2.73), while the maximum hydro power flow at Sutami is about 132 m^3/s, so there is not much room to manoeuver and some spill is inevitable. This is why there is no large variation of connection tunnel flow in the first time step, however the next 36 time steps are different. Therefore, each row of 925 numbers contains enough information to rebuild one complete feasible solution. The first number in each row identified as the "Penalty" is the actual value of the objective function (9.20) as calculated by the model. Note that this value can be either positive ore negative, and that the goal is to obtain the minimum possible value. The initial 10 members of the mating pool have been improved on average by only 30% after 1000 more members were generated in a completely random manner. Additional 4000 random members improve the quality of the mating pool by about 45% compared to the initial 15 members, but after that any additional improvement would be very small and

Table 9.1 Objective function for various stages of SFEP progression

The first 10 members of the mating pool

-- Penalty	---[131]	----[72]	----[73]	----[61]-
547501367.521	4.75	2.73	0.00	132.39
559135044.575	4.64	2.84	0.00	132.08
660889810.337	4.64	2.84	0.00	132.08
670514728.533	4.64	2.84	0.00	132.08
683568706.346	4.47	3.01	0.00	131.57
693739724.549	4.47	3.01	0.00	131.57
702861279.333	4.64	2.84	0.00	132.08
724973698.278	4.64	2.84	0.00	132.08
745785780.547	4.64	2.84	0.00	132.08
773558991.706	4.47	3.01	0.00	136.63

Mating Pool members after initialization procedure resulting in 1000 randomly created individuals

-- Penalty	---[131]	----[72]	----[73]	----[61]-
427198566.742	4.64	2.84	0.00	132.08
443901576.517	4.47	3.01	0.00	131.57
468538947.104	4.47	3.01	0.00	131.57
471216857.657	4.47	3.01	0.00	131.57
473362569.525	4.64	2.84	0.00	132.08
478902846.830	4.57	2.91	0.00	131.86
479941884.497	4.64	2.84	0.00	132.08
499612370.646	4.47	3.01	0.00	133.00
499621437.714	4.47	3.01	0.00	134.76
503946079.600	4.47	3.01	0.00	131.57

Mating Pool after the first 10 recombinations

-- Penalty	---[131]	----[72]	----[73]	----[61]-
38145555.645	4.57	2.91	0.00	131.85
349728097.574	4.47	3.01	0.00	131.57
356201935.008	4.47	3.01	0.00	131.57
378282973.290	4.57	2.91	0.00	131.85

424806143.423	4.64	2.84	0.00	132.08
427198566.742	4.64	2.84	0.00	132.08
437609286.822	4.49	2.99	0.00	131.61
443901576.517	4.47	3.01	0.00	131.57
447996201.007	4.47	3.01	0.00	131.57
468538947.104	4.47	3.01	0.00	131.57

Mating Pool after 20 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-53184871.781	4.47	3.01	0.00	131.57
-6952527.701	4.47	3.01	0.00	131.57
38145555.645	4.57	2.91	0.00	131.85
300866705.501	4.51	2.97	0.00	131.65
303438246.571	4.57	2.91	0.00	131.85
349728097.574	4.47	3.01	0.00	131.57
356201935.008	4.47	3.01	0.00	131.57
378282973.290	4.57	2.91	0.00	131.85
410797683.536	4.53	2.95	0.00	131.74
422141278.291	4.47	3.01	0.00	131.57

Mating Pool after 30 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-53184871.781	4.47	3.01	0.00	131.57
-20221171.995	4.57	2.91	0.00	131.85
-12657188.718	4.47	3.01	0.00	131.57
-6952527.701	4.47	3.01	0.00	131.57
6154141.700	4.47	3.01	0.00	131.57
21927448.248	4.57	2.91	0.00	131.85
23135214.139	4.47	3.01	0.00	131.57
38145555.645	4.57	2.91	0.00	131.85
83895665.813	4.47	3.01	0.00	131.57
211459134.476	4.57	2.91	0.00	131.85

Mating Pool after 40 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-53184871.781	4.47	3.01	0.00	131.57
-52653475.505	4.51	2.97	0.00	131.69
-38911474.187	4.57	2.91	0.00	131.85
-35819246.664	4.48	3.00	0.00	131.60
-29851943.183	4.49	2.99	0.00	131.62
-20221171.995	4.57	2.91	0.00	131.85
-15289749.867	4.48	3.00	0.00	131.60
-12657188.718	4.47	3.01	0.00	131.57
-12069957.783	4.57	2.91	0.00	131.85
-6952527.701	4.47	3.01	0.00	131.57

Mating Pool after 50 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-59880515.731	4.57	2.91	0.00	131.85
-58610365.734	4.49	2.99	0.00	131.64
-53184871.781	4.47	3.01	0.00	131.57
-52653475.505	4.51	2.97	0.00	131.69
-50644168.650	4.57	2.91	0.00	131.84
-42471090.880	4.57	2.91	0.00	131.85
-39522745.687	4.50	2.98	0.00	131.67
-38911474.187	4.57	2.91	0.00	131.85
-38063458.027	4.48	3.00	0.00	131.59
-35819246.664	4.48	3.00	0.00	131.60

Mating Pool after 100 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-63268689.809	4.50	2.98	0.00	131.65
-62395074.722	4.57	2.91	0.00	131.85
-61696592.084	4.50	2.98	0.00	131.65

-60801009.577	4.50	2.98	0.00	131.65
-59880515.731	4.57	2.91	0.00	131.85
-58993359.880	4.50	2.98	0.00	131.65
-58610365.734	4.49	2.99	0.00	131.64
-58092264.331	4.51	2.97	0.00	131.68
-57127295.450	4.57	2.91	0.00	131.87
-53369495.251	4.50	2.98	0.00	131.65

Mating Pool after 150 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-67027274.821	4.50	2.98	0.00	131.65
-66958659.611	4.50	2.98	0.00	131.65
-66639419.358	4.48	3.00	0.00	131.60
-66581659.445	4.57	2.91	0.00	131.87
-66490324.241	4.47	3.01	0.00	131.57
-66466367.861	4.50	2.98	0.00	131.66
-66168593.032	4.57	2.91	0.00	131.87
-65068900.831	4.51	2.97	0.00	131.69
-64712454.207	4.48	3.00	0.00	131.59
-64622448.540	4.57	2.91	0.00	131.87

Mating Pool after 200 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-67788538.134	4.51	2.97	0.00	131.70
-67373646.137	4.64	2.84	0.00	132.08
-67243249.961	4.64	2.84	0.00	132.08
-67142915.424	4.51	2.97	0.00	131.69
-67027552.382	4.50	2.98	0.00	131.65
-67027274.821	4.50	2.98	0.00	131.65
-67011366.233	4.47	3.01	0.00	131.57
-66958659.611	4.50	2.98	0.00	131.65
-66664978.566	4.51	2.97	0.00	131.69
-66639419.358	4.48	3.00	0.00	131.60

Mating Pool after 300 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-71088013.412	4.51	2.97	0.00	131.69
-70843144.402	4.53	2.95	0.00	131.74
-70044911.699	4.49	2.99	0.00	131.61
-70038755.226	4.52	2.96	0.00	131.70
-69949546.150	4.53	2.95	0.00	131.74
-69312838.383	4.52	2.96	0.00	131.69
-69111450.715	4.53	2.95	0.00	131.74
-69081912.797	4.53	2.95	0.00	131.74
-68815832.367	4.53	2.95	0.00	131.74
-68758825.843	4.53	2.95	0.00	131.74

Mating Pool after 500 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-72495075.526	4.49	2.99	0.00	131.62
-72029875.865	4.47	3.01	0.00	131.57
-72023906.988	4.48	3.00	0.00	131.59
-71977750.082	4.53	2.95	0.00	131.75
-71970862.692	4.49	2.99	0.00	131.61
-71925229.395	4.47	3.01	0.00	131.57
-71800594.002	4.53	2.95	0.00	131.74
-71796512.238	4.48	3.00	0.00	131.59
-71786575.267	4.50	2.98	0.00	131.64
-71785525.034	4.48	3.00	0.00	131.59

Mating Pool after 1000 recombinations

-- Penalty ---	[131]	[72]	[73]	[61]
-74451941.077	4.48	3.00	0.00	131.60
-74367166.629	4.48	3.00	0.00	131.60

-74360959.943	4.48	3.00	0.00	131.60
-74291170.388	4.48	3.00	0.00	131.60
-74288269.285	4.48	3.00	0.00	131.60
-74288045.782	4.48	3.00	0.00	131.60
-74282711.601	4.48	3.00	0.00	131.60
-74275993.082	4.48	3.00	0.00	131.60
-74252361.960	4.51	2.97	0.00	131.68
-74237789.123	4.49	2.99	0.00	131.61

it would come at a great cost in CPU time. Yet genetic recombination operator speeds up the improvements tremendously. After starting the process of recombination, of the first 10 individuals generated using the cross-over technique, 7 were eligible for inclusion in the mating pool. In the next 20 recombinations the model managed to find 2 solutions which do not violate the maintenance flow target and provide higher reservoir levels, such that huge positive penalties have disappeared and the negative values (indicating benefit to PJT) have prevailed. Again, only 4 parents have survived the transition from the tenth to the twentieth recombination, six of them were replaced in the mating pool by their children. This can be identified by comparing the values of the objective functions in the mating pool after the first 10 recombinations and after 20 recombinations. A high quality parent solution, such as the top ranked parent after 20 recombinations with the objective function of -5318487.781 will offer its genetic material to almost 100 children before it is pushed out of the mating pool by new solutions with better fitness. After completing the first 1000 recombinations, there is very little improvement. Most independent decision variables have converged and further progress is only possible by mutation, which was setup as a small normalized change to about 2% of all independent decision variables, such that it plays a minor role in this problem. By comparison, solution of individual time steps of Case A resulted in the total value of the objective function for the whole year of -41743714 Rp (negative value representing benefit due to minimization of the objective function), almost 50% below the -74237789 Rp obtained for annual optimization in Case B, which was to be expected due to having a perfect inflow forecast in Case B..

It takes roughly 1 second of CPU time on a 500 MHz PC to generate 100 solutions. The above simulation run has a total of 2000 generated solutions which takes about 20 seconds. Included in the solution process is a sophisticated iteration scheme which solves the reservoir

balance for Lahor and Sutami reservoirs several times iteratively until the connection tunnel and water balance constraints are all satisfied. This iterative scheme reduces the variance of the initially generated guess for the connection tunnel flow within the feasible realm, which is why large variation of the values is not apparent in Table 9.1 even at the initial stage.

Both objective functions have flaws that must be dealt with. Using a single time step optimization in an effort to maximize hydro power results in an empty reservoir within several time intervals. The reservoir then does not re-fill for the rest of the year since hydro power has higher cost factor than storage. Therefore, it was concluded that Case A cannot be run without including some kind of reservoir operating guidelines, such as the operating zones used in Case Study II or a realistic hydro power target for each time interval. Typically, Case B should be run first and its results analyzed in an effort to try to generate some the shape of storage zones and their pricing vectors which work the best for short term forecasts considered in Case A. This effort was beyond the scope of testing the SFEP solver.

The danger of finding the best overall solution using the objective function as in Case B results in an interesting phenomena – the model decides to empty storage at the end of the year and thus increase power production. To counter this, an additional time interval was added to include the first 10 days of the following year and place the minimum power generating elevation of 260 m at Sutami at the end of 37^{th} time interval, which is then discarded in the final analyses. This worked in some cases, but it still resulted in reservoir levels that were below the full supply level at the end of the year, when the wet season has usually been in full swing for at least a month. The effect of the connection tunnel routine on reservoir levels is discussed below.

Objective function B has been applied to the Brantas Basin in all 23 years (1977 to 1999). In comparison to the historic operation, it provided higher power generation in each year, and on average by about 6.5% in spite of the high maintenance flow requirement of 20 m^3/s which was always met in the model while in reality the minimum target was only 7 m^3/s.

Higher maintenance flow requirement has caused an increase in irrigation deficits in the Delta region on average by about 3.97%.

There is a hydro power plant only downstream of Sutami, so any spills from Lahor are a loss to power generation at Sutami. The operators are then faced with an enviable task of having to generate enough head between the two reservoirs to minimize spills without having a reliable inflow forecast. Consider for example simulated results for year 1992. This was a wet year, so the ideal should be to keep both reservoirs full (i.e. Lahor elevation at 272.7 m and Sutami at 272.5 m). This means that the available head is only 0.2 meters. According to (9.11) and (9.12) this corresponds to a tunnel flow of 4.47 m^3/s. Connection tunnel flow of 15 m^3/s corresponds to about 2.25 m average head differential between the two reservoirs over a time step. Assume the following situation: both reservoirs are full, Lahor inflow is 15 m^3/s and the reservoir operators want to send 15 m^3/s of Lahor inflow into Sutami via the connection tunnel (this results in no spills from Lahor). To obtain 15 m^3/s of tunnel flow the elevation of Sutami should be reduced from 272.5 to 268.4 over a given time step. Lahor elevation would remain the same since its inflow equals outflow (in this case all outflow is placed in the connection tunnel). Therefore, the average elevation at Lahor is 272.7, while the average elevation at Sutami is

$$(272.5+268.4)/2 = 270.45$$

This gives the difference between the average elevation of Lahor and Sutami of 272.7 - 270.45 = 2.25 m, which provides the necessary head differential to rout a flow of 15 m^3/s through the tunnel. This also means the loss of head for power generation of Sutami in the order of 2.25 m. Inflows will change in subsequent time steps and the operators must try to maintain levels of both reservoirs close, while in the same time trying to minimize the spill from both reservoir that would bypass hydro power plant at Sutami.

A manual check was conducted to verify that the tunnel flows calculated as a function of the

Table 9.2 Tunnel flows from the model and from direct calculation

Year 1993 10-day ending	Lahor 10-day ending elevation	Sutami 10-day ending elevation	Connection tunnel flows from SFEP (m3/s)	Connection tunnel flows using hydraulic formula (m3/s)
initial elevation	270.770	266.120		
1	271.697	268.018	20.41	20.41
2	272.700	268.814	19.44	19.45
3	272.700	270.677	17.18	17.19
4	272.700	271.171	13.33	13.33
5	272.627	271.823	10.80	10.80
6	272.700	272.162	8.19	8.19
7	272.700	272.498	6.08	6.08
8	272.700	269.889	12.27	12.27
9	272.066	268.665	17.62	17.62
10	272.111	267.785	19.65	19.66
11	272.602	272.490	14.90	14.90
12	272.700	271.943	6.59	6.59
13	271.166	268.034	13.95	13.94
14	268.862	268.959	12.32	12.32
15	269.359	268.846	4.56	4.56
16	271.732	272.144	2.26	2.26
17	272.700	272.062	3.37	3.37
18	272.260	272.190	5.95	5.95
19	272.342	272.149	3.63	3.63
20	272.481	272.459	3.28	3.28
21	272.580	272.385	3.30	3.30
22	272.559	272.500	3.56	3.57
23	272.646	272.479	3.36	3.36
24	272.572	272.425	3.96	3.96
25	272.089	271.670	5.32	5.32
26	272.064	272.184	3.87	3.87
27	272.368	272.099	2.73	2.73
28	267.611	263.353	15.05	15.05
29	262.672	264.432	11.18	11.18
30	265.388	263.657	-1.19	-1.19
31	262.915	264.029	5.56	5.56
32	265.147	263.805	3.38	3.38
33	264.122	264.172	8.04	8.04
34	265.670	264.526	7.40	7.40
35	262.610	261.498	10.62	10.62
36	261.632	261.775	6.96	6.96

average reservoir elevations at Sutami and Lahor is within 0.01 m^3/s of the value obtained by the SFEP simulation. This was followed by a check of the reservoir balance for Sutami and Lahor, since the tunnel flow is a constituent of reservoir balance for both reservoirs. In column 4 of Table 9.2 the tunnel flows were assigned by the SFEP, while in column 5 they are calculated as function of the average head differential between the two reservoirs. A table similar to Table 9.2 can be generated for each simulated year. Figure 9.4 shows the elevation of Lahor and Sutami for all 23 years using objective function Case B.

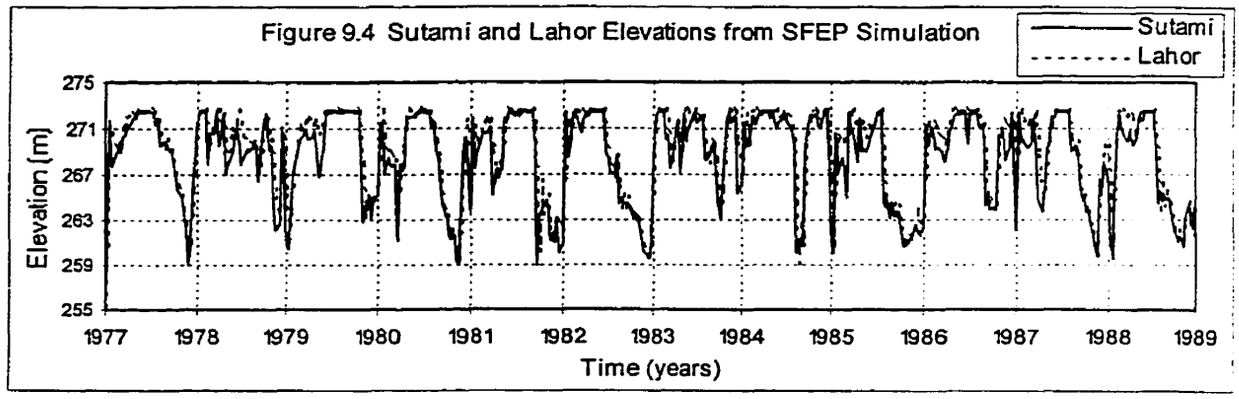
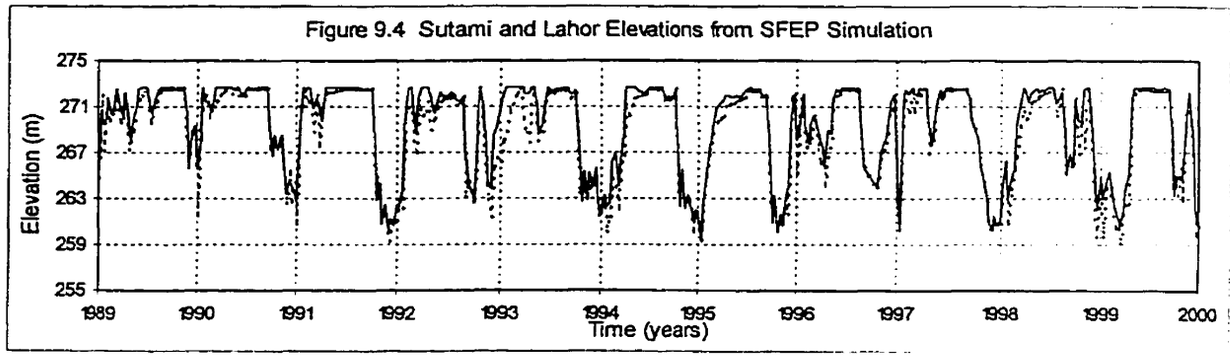


Figure 9.4 Sutami and Lahor Elevations from SFEP Simulation


The elevations of the two reservoirs are closely tied together, which should be expected as a result of the connection tunnel. Table 9.2 demonstrates that the connection tunnel flows comply with difference between the two reservoirs. Large inflows into Lahor cause lowering of the elevation at Sutami reservoir in order to create enough head differential to route most of Lahor inflows through the tunnel. This in turn reduced the head of Sutami reservoir, but

it also increases the net head of Sungguruh hydro power plant located upstream of Sutami dam. This kind of constraint could not be handled using other models based on linear programming.

PJT indicated a strong desire to use this model as a planning tool to study the various cost factors as part of the on-going negotiation if the new water pricing policy with the Provincial Government of East Java.

9.6 Conclusions

The Brantas river basin offers a typical size water allocation problem with several common non-linear constraints related to hydro power generation, and a few unique non-linear constraints related to the connection tunnel flow. The SFEP solver was modified to include several iterative steps between solving the balance equation at Lahor and Sutami reservoir such that both the connection tunnel flow formula (9.17) and both reservoir balance equations are satisfied. These iterative steps are also a form of "gene therapy" approach, but with additional sophistication since two reservoirs and their levels are involved, along with the dynamic maximum turbine flows from Sutami reservoir. The SFEP managed to keep all generated solution within the feasible domain and progressed to the favorable regions of the search space rather quickly.

The only downside in solving this problem for the whole year (Case B) is that it seems that an annual solution must be obtained with one or more time intervals of the following year included in the simulation, to avoid having the ending reservoir level of annual simulation at the minimum operating level for its respective hydro power plant. These time intervals belong to the following year and they can bias the solution in the given year if the inflows are excessively dry or wet.

10 CONCLUSIONS

10.1 Summary of the SFEP Features

This research attempted to develop a specialized type of genetic algorithm suitable for solving non-linear network flow programs which may be non-linear both in terms of bounds and the objective function. The main features of the approach are the massive initialization procedure that initiates the search from many feasible points in the search space simultaneously, and recombination procedure which maintains feasibility of offspring by employing the gene therapy operator. The search can also be assisted by using heuristic rules that generate more individuals in the favorable areas of the search space when such areas can be determined. The algorithm provides a stable and relatively fast convergence when compared to other similar search methods, mainly due to restricting the search within the feasible region. This helps reduce the large overhead associated with generating infeasible solutions which is common to other EP search methods.

The proposed algorithm was first tested on a series of non-linear transportation problems for which various high quality solutions were available in the literature. In addition, the linear case was added and its solution compared to a standard linear TP solver. The proposed algorithm either equaled or surpassed the previous solutions in terms of the quality of the objective function, and it proved to be over ten times faster for several cases of complex objective functions with multiple minima. The test problems ranged between 49 and 100 decision variables.

The second set of tests was a water resources network with unusual constraints related to the sum of hydro power being equal to a specified power target. The main difficulty in this problem was that this constraint was not a direct function of flow. Rather, it is a function of all 52 inflows, the starting reservoir levels at the beginning of the year and the reservoir outflows at each of the time steps preceding a given time step. This constraint could not be

included in the model such that the model ensures the search within the feasible region alone. Hence, a high penalty term had to be added for violating this constraint. Although this seems like a departure from the stated goal to search only within the feasible region, the final results obtained in the study seem very optimistic.

The final test runs were conducted on the Brantas river basin modeling schematics. This problem is of standard size for many water resources studies and it has all components that typically play a role in studies of this size. The objective function is non-linear, and the constraints related to hydro power and the connection tunnel are also non-linear. This problem required additional work on modifying and expanding the gene therapy concept to include difficult hydraulic constraint related to the connection tunnel flows. The model provides fast solutions however the period being optimized has to be extended by a few additional time intervals that should be discarded in the final analyses.

10.2 Future Research Directions

There are several possible improvements to this approach that require a larger study framework. At this point the user has to "calibrate" the model to a particular problem by finding the most efficient size (number of parents) of the mating pool, the optimal number of individuals generated during initialization as well as the best mutation frequency and mechanism. These issues can be resolved by building a database of previously solved networks and recording their size and complexity along with the best parameters that were found for a particular application. The solver should eventually be able to scan a new problem in terms of its size, the shape of the objective function and the complexity of the constraints, try several parameter options and record the one that works the best such that on similar problems in the future it can refer to the best set of parameters that were determined on earlier runs. This means that the SFEP solver can be programmed to be self-adaptive to a various size and types of network flow problems. The process of developing self-adapting mechanism must be coupled with building a database of parameters used in the previous

solutions that the model could access and "learn" from them. This undertaking was beyond the scope of this research.

For example, when the objective function is clearly defined for each variable, as in the case of the transportation problem example, the solver could inspect the values in each cell by solving the objective function for that cell for 100 discrete values of the argument. The values with the best fitness (i.e. the local minimums) would then be placed in the database and used in the initialization procedure to generate more solutions in the favorable areas of the search space.

For each application the user has to decide which arcs form the maximum spanning tree. This means that the user must know what a given network represents and what are the decision variables in it. Similar to other heuristic search techniques, an application of this algorithm is problem specific, and the knowledge about the problem being solved is essential. An expert system encompassing a database of previous successful applications would help a novice user, but the final success of this kind of tool is still very much dependent on the proper setup, which depends on the experience and judgement of the user.

11 REFERENCES

Alberta Environment. 1995. *Water Resources Management Model (WRMM) User's Manual.* Calgary, Alberta.

Ahuja, R.K, Goldberg, A.V., Orlin, J.B. and Tarjan, R.E. 1992. Finding minimum cost flow by double scaling. *Mathematical Programming, 53, p. 243-266.*

Ahuja, R.K, Magnanti T.L., Orlin, J.B. (1993). *Network Flows*. Prentice Hall.

Andrews, E.S., F.I. Chung and J.B. Orlin 1993. Multilayers, priority-based simulation of conjunctive facilities. *Journal of Water Resources Planning and Management, ASCE, 118(1), p. 32-53.*

Avriel, M. 1976. *Nonlinear Programming: Analysis and Methods*. Prentice Hall, Englewood Cliffs, NJ.

Back, T., F. Hoffmeister and H.P. Schewel. 1991. A survey of evolution strategies. *Proceedings of The Fouth International Conference on Genetic Algorithms.* R.K. Belew and L.B. Booker (Eds),2. Morgan Kaufmann, San Mateo, CA.

Back, T. and H.P. Schewel. 1993. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation 1(1), p. 1-23.* The Massachusetts Institute of Technology.

Barr, R.S., F. Glover, and D. Klingman. 1974. An improved version of the out-of-kilter method and comparative study of computer codes. *Mathematical Programming., 7, p. 60-86.*

Barahona, F., and E. Tardos. 1989. Note on Weintraub's minimum cost circulation algorithm.

SIAM Journal of Computing, 18, p. 579-583.

Bellman, R.E. 1957. *Dynamic Programming.* Princeton University Press, N.J.

Bertsekas, D.P. 1979. A distributed algorithm for the assignment problem. *Working Paper, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.*

Bertsekas, D.P. 1987. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice Hall, Englewood Cliffs, NJ.

Bertsekas, D.P., and P. Tseng. 1988. Relaxation methods for minimum cost ordinary and generalized network flow problems. *Operation Research, 36, p. 93-114.*

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

Bijaya, P.S., L. Duckstein & E. Z. Stakhiv. 1996. Fuzzy rule-based modeling of reservoir operation. *Journal of Water Resources Planning and Management, ASCE, 122(4).*

Bland, R.G. and D.L. Jensen. 1992. On the computational behaviour of a polynomial-time network flow algorithm. *Mathematical Programming, 54, p. 1-39.*

Box, M.J., D. Davies and W.H. Swann. 1969. Nonlinear optimization techniques. *ICI Monograph 5, Oliver and Boyd, Edinburgh.*

Bradley, G., G. Brown and G. Graves. 1977. Design and Implementation of large scale primal transhipment algorithms. *Management Science., 21, p. 1-38.*

Brendecke, C.M. 1989. Network models of water rights and system operations. *Journal*

of Water Resources Planning and Management, ASCE, 115(5), p. 684-696.

Brooke, A., D. Kendrik, and A. Meeraus. 1996. *GAMS: A User's Guide.* The Scientific Press, Redwood City, California.

Brown, R.R. 1959. A generalized computer procedure for the design of optimum systems. *AIEE Transaction I, Computational Electronics 78, p. 285-293.*

Brown, G.G. and R.D. McBride. 1984. Solving generalized networks. *Management Science 30(12), p. 1947-1523.*

Brown, G.G., R.D. McBride, and R.K. Wood. 1985. Extracting embedded generalized networks from linear programming problems. *Mathematical Programming, 21, p. 11-31.*

Brown, K.M., and J.E. Dennis Jr. 1972. Derivative free analogues of the levenberg-marquardt and Gauss algorithms for nonlinear least squares approximation. *Numerical Mathematics 18, p. 289-297.*

Busacker, R.G. and P.J. Gowen. 1961. A procedure for determining minimal-cost network flow patterns. *ORO Technical Report 15, Operational Research Office, John Hopkins University, Baltimore, MD.*

Busacker, R.G. and T.L. Saaty. 1965. *Finite Graphs and Networks.* McGraw Hill, New York.

Chung, F.I., M.C. Archer & J.J. DeVries 1989. Network flow algorithm applied to California aqueduct simulation. *Journal of Water Resources Planning and Management, ASCE, 115(2), p. 131-147.*

Curry, H.B. 1944. The method of steepest descent for nonlinear minimization problems.

Quarterly Applied Mathematics 21, p. 258-261.

Dantzig, G.B. 1963. *Linear Programming and Extension.* Princeton University Press, Princeton, NJ.

Davis, L. (Editor). 1987. *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, San Mateo, CA.

Davis, L. 1991. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

Denardo, E. 1982. *Dynamic Programming Theory and Applications.* Prentice Hall, Englewood Cliffs, NJ.

De Jong, K.A. 1975. *An analysis of the behaviour of a class of genetic adaptive systems.* Ph.D. Thesis, University of Michigan. Dissertation Abstract International, 36(10), 5140(B).

De Jong, K.A. 1988. Learning with genetic algorithm: an overview. *Machine Learning, 3 (2), p. 121-138.*

De Jong, K.A. 1990. Using genetic algorithms for supervised concept learning. *Proceedings of the Second International Conference on Tools for AI.*

Doris R. R. and S. Chen. 1981. A comparison of three algorithms for finding fundamental cycles in a directed graph. *Networks 11, p. 1-12.*

Edmonds, J., and R.M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of ACM, 19, p. 248-264.*

Elam, J., F. Glover, and D. Klingman. 1979. A strongly convergent primal simplex algorithm

for generalized networks. *Mathematics of Operations Research, 4, p. 39-59.*

Evanson, D.E. & J.C. Moseley. 1970. Simulation/optimization techniques for multi-basin water resources planning. *Water Resources Bulletin, 6(5), p. 725-736.*

Fletcher, R., and C.M. Reeves. 1964. Function minimization by conjugate gradients. *Journal of Computing 7, p. 149-154.*

Fletcher, R., and M.J.D. Powell. 1963. A rapid convergence descent method for minimization. *Journal of Computing 6, p. 163-168.*

Fogel, L.J., Owens, A.J., and Walsh, M.J. 1966. *Artificial Intelligence Through Simulated Evolution*, John Wiley, Chichester, UK.

Ford, L.R., and D.R. Fulkerson. 1962. *Flows in Networks*. Princeton University Press, Princeton, NJ.

Forrest, S. 1993. *Proceedings of the Fifth International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA.

Fulkerson, D.R. 1961. An out-of-kilter method for minimal cost flow problems. *SIAM Journal on Applied Mathematics, Vol. (9), No. 1.*

Gill, P.E., and W. Murray. 1972. Quasi-Newton methods for unconstrained optimization. *Journal of Industrial and Applied Mathematics 9, p. 91-108.*

Glover, F. 1987. Genetic algorithms and scatter search: unsuspected potentials. *Statistic and Computing, Vol. 4.*

Glover, F., D. Karney and D. Klingman. 1974. Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problem. *Networks, 4, p. 191-212.*

Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA.

Glover F. 1999. Scatter Search and Path Relinking. *To appear in New Methods of Optimization.* McGraw Hill.

Grafenstette, J.J. 1987. Incorporating problem specific knowledge into genetic algorithms. in *Genetic Algorithms and Simulated Annealing, p. 42-60,* Morgan Kaufmann Publishers, San Mateo, CA.

Gunter, R. 1994. Massively parallel simulated annealing and its relation to evolutionary algorithms. *Evolutionary Computation, Vol. 1, (4), p. 361-383.*

Goldberg, A.V. and R.E. Tarjan. 1987. *Solving Minimum Cost Problem by Successive Approximation. Journal of ACM 35(1988), p. 921-940.*

Goldberg, A.V. and R.E. Tarjan. 1988. Finding minimum cost circulations by cancelling negative cycles. *Mathematics of Operations Research 15 (1990), p. 430-466.*

Goldberg, D. 1987. Genetic algorithms in pipeline optimization. *Journal of Computing in Civil Engineering, Vol. 9, No.1.*

Golden, B.L. and T.L. Magnanti. 1977. *Deterministic Network Optimization: A Bibliography.* John Wiley & Sons Inc.

Goldstein, A.A. 1962. Cauchy's method of minimization. *Numerical Mathematics 4, p. 146-150.*

Goldstein, A.A. 1965. On Newton's Methods. *Numerical Mathematics 7, p. 391-393.*

Grefenstette, J.J. 1987. *Proceedings of the Second International Conference on Genetic Algorithms.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Hillier, F.S., and G.J. Lieberman. 1995. *Introduction to Operations Research.* McGraw Hill.

Holland, J.H. 1975. *Adaption in Natural and Artificial Systems.* University f Michigan Press, Ann Arbor, Michigan.

Hooke, R., and T.A. Jeeves. 1961. Direct search solution of numerical and statistical problems. *Journal of Applied Computational Methods 8, p. 212-229.*

Householder, A.S. 1953. *Principles of Numerical Analysis.* McGraw Hill, New York.

Hunter, A. 1998. Crossing over genetic algorithms: the sugal generalised GA, *Journal of Heuristics, Vol. 4, p. 179-192.*

Ilich, N. and S.P. Simonovic. 1998. Evolutionary Algorithm for Minimization of Pumping Costs. *ASCE Journal of Computing, Vol. 12, No.4.*

Ilich, N. 1993. Improvement of the return flow allocation in the water resources management model (WRMM) of Alberta Environment. *Canadian Journal of Civil Engineering, Vol. 20, No. (4).*

Ilich, N. and S.P. Simonovic. 2000 (a). The benefits of computerized real-time river basin

management in the Malahayu reservoir system. *Canadian Journal of Civil Engineering. Vol. 27, No. 1., p. 55-64.*

Ilich, N., R. Drury, P. Godman and S. Simonovic. 2000 (b). Operation of Bighorn/Brazeau hydro power system of Transalta Utilities Corporation. *Proceedings of the XIII International Conference on Computational Methods in Water Resources, Vol. 2, p. 991-998.* A.A. Balkema Publishers, Rotterdam.

Ilich, N. and S.P. Simonovic. 2001. An evolution program for non-linear transportation problems. *To be published in the Journal of Heuristics.* Kluwer Academic Publishers, March 2001.

Iri, M. 1960. A new method of solving transportation network problems. *Journal of the Operation Research Society of Japan, 3, 27-87.*

Jacoby, S.L.S., J.S. Kowalik, and J.T. Pizzo. 1972. *Iterative Methods for Nonlinear Optimization Problems.* Prentice Hall, Englewood Cliffs, NJ.

Janikow, C. and Z. Michalewicz. 1990. Specialized genetic algorithms for numerical optimization problems. *Proceedings of the International Conference on Tools for AI,* 798-804.

Jensen, P.A., and W. Barnes. 1980. *Network Flow Programming.* Wiley, New York.

Jewell, W.S. 1958. Optimal flow through network. *Interim Technical Report 8, Operational Research Center, MIT, Cambridge, MA.*

Kantorovich, L.V. 1945. On an effective method of solving extremal problems for aquadratic functionals. *Compt. Rend. Acad. Sci. URSS. 48, 455-460.*

Kennington, J.L., and R.V. Helgason. 1980. *Algorithms for Network Programming*. Wiley-Intersence, New York.

Kirkpatrick, S., C. Gelatt, and M. Vecchi. 1983. *Optimization by Simulated Annealing*. Science.

Klein, M. 1967. A primal method for minimal cost flows with application to the assignment and transportation problems. *Management Science, 14, 205-220.*

Koza, J. 1993. *Genetic Programming: on the programming of computers by means of natural selection.* MIT Press.

Kowalik, G.A., and M.R. Osborne. 1968. *Methods for Unconstrained Optimization Problems*. Amer. Elsievier, New York.

Kuhn, H.W. and A.W. Tucker. 1951. Nonlinear Programming, in Jerzy Neyman (ed.), Proceedings of the Second Berkley Simposium, *University of California Press, Berkley, 1951, p. 481-492.*

Koza, J.R. (1992). *Genetic Programming*, MIT Press, Cambridge, MA.

Kuczera, G. and G. Diment. 1988. General water supply system simulation model: WASP. *Journal of Water Resources Planning and Management, ASCE, 114(4), p. 365-382.*

Labadie, J.W., DA. Bode, and A.M. Pineda. 1986. Network model for decision-support in municipal raw water supply. *Water Resources Bulletin, Vol. 22, No. 6, p. 927-940.*

Lansley, K., and L. Mays. 1989. Optimization model for water distribution system design.

Journal of Hydraulics, Vol. 115, No. 10.

Levenberg, K. 1944. A method for the solution of certain nonlinear problems in least squares. *Quarterly Applied Mathematics, Vol 2, p. 164-168.*

Loucks, D.P., J.R. Stedinger J.R. and D.A.Haith 1981. *Water Resources Systems Planning and Analysis.* Prentice Hall Inc., Englewood Cliffs, N.J.

McBride, R.D. 1985. Solving embedded generalized network problems. *European Journal of Operational Research 21, p. 82-92.*

McCormick, G.P. 1983. *Nonlinear Programming: Theory, Algorithms and Applications.* Wiley, New York.

Michalewicz, Z. 1994. *Genetic Algorithms + Data Structures = Evolutionary Programs*, Springer-Verlag, KG, Berlin, Germany.

Michalewicz, Z., Janikow C., and Krawczyk J. 1992. A modified genetic algorithm for optimal control problems, *Computers and Mathematics with Applications, 23 (12) 83-94.*

Michalewicz, Z., Vignaux, G.A. and Hobbs, M. 1991. A non-standard genetic algorithm for the non-linear transportation problem. *ORSA Journal of Computing, Vol. 3, No. 4, p. 307-316.*

Minty, G.J. 1960. Monotone Networks. *Proceedings of the Royal Society of London, 257A, p. 194-212.*

Murty, K.G. 1992. *Network Programming.* Prentice Hall.

Nelder, J.A. and R. Mead. 1965. A simplex method for function minimization. *Journal of Computing Vol. 7, p. 308-313.*

Oliviera, R. and D.P. Loucks. 1997. Operating rules for multi reservoir systems. *Water Resources Research, Vol. 33, No. 4, p. 839-852.*

Orlin, J.B. 1988. A faster strongly polynomial minimum cost flow algorithm. *Proceedings of the 20th ACM Symposium on the Theory of Computing, p. 377-387.*

Ortega, J. M., and W.C. Rheinboldt . 1967. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York.

Ostrowski, A.M. 1967. Contributions to the theory of the method of steepest descent. *Archive of Rational Mechanics Annals 26, p. 257-280.*

Powell, M.J.D. 1962. An iterative method for finding stationary values of a function of several variables. *Journal of Computing 5, p. 147-151.*

Rechenberg I. (1965). A Cybernetic Solution Path of an Experimental Problem. *Royal Aircraft Establishment, translation No. 1122, Ministry of Aviation, Farnborough Hants, UK.*

Rockafellar, R. T. 1984. *Network Flows and Monotropic Optimization.* John Wiley & Sons.

Rock, H. 1980. Scaling techniques for minimal cost network flows. *In Discrete Structures and Algorithms. Edited by V. Page. Carl Hanser, Munich, p. 181-191.*

Rosenbrock, HH. 1960. An automatic method for finding the greatest or least value of a function. *Journal of Computing 3, p. 175-184.*

Rudolph, G. 1994. *Massively Parallel Simulated Annealing and Its Relation to Evolutionary Algorithms.* MIT Press.

Russell, S.O. & P.F. Campbell 1996. Reservoir operating rules with fuzzy programming. *Journal of Water Resources Planning and Management, ASCE, 122(4), p. 165-170.*

Schechter, S. 1968. Relaxation Methods for Convex Problems. *SIAM Journal of Numerical Analyses 5, p. 601-612.*

Sigvaldason, O.T. 1976. A simulation model for operating a multipurpose multireservoir System. *Water Resources Research, 12(2), p. 263-278.*

Simonovic, S.P. 1987. The implicit stochastic model for reservoir yield optimization. *Water Resources Research, 23(12), p. 2159-2167.*

Sniedovich, M. 1991. *Dynamic Programming.* Marcel Dekker, New York.

Sun, Y-H., W. W-G. Yeh, N-S. Hsu, and P.W.F. Louie. 1995. Generalized Network Algorithm for Water-Supply-System Optimization. *Journal of Water Resources Planning and Management, ASCE, 121(5), p. 392-398.*

Swann, W.H. 1964. Report on development of a new direct searching method of optimization. *ICI, Centr. Instr. Lab., research note 64-3, Middlesborough, Yorks.*

Tucker, A.W. 1963. Combinatorial theory underlying linear programs. In *Recent Advances in Math Programming. R.L.* R.L. Graves and P. Wolfe, eds. McGraw Hill.

Vignaux, G.A. and Michalewicz, Z. 1989. A genetic algorithm for the transportation problem. *Proceedings of the Fourth International Symposium on Methodologies for*

Intelligent Systems, p. 252-259, North Holland, Amsterdam.

Wallacher, C. and U.T. Zimmermann. 1991. A combinatorial interior point method for network flow problems. *Fourteenth International Symposium on Mathematical Programming, Amsterdam, The Netherlands.*

Wegge, L. 1966. On a discrete version of the Newton-Raphson method. *SIAM Journal of Numerical Analyses 4, p. 134-142.*

Weintraub, A. 1974. A primal algorithm to solve network flow problems with convex cost. *Management Science, 21, p. 87-97.*

Whitley, D., 1989. The Genitor algorithm for selective pressure: why rank-based allocation of reproductive trials is the best. *Proceedings of the Third International Conference On Genetic Algorithms.* Morgan Kaufmann.

Wolfe, P. 1969. Convergence conditions for ascent methods. *SIAM Review 11, p. 226-235.*

Wolfe, P. 1970. Convergence theory in nonlinear programming. *Editor Abadie (Integer and Non Linear Programming), North Holland Publishing, p. 1-36.*

Wolfe, P. 1971. Convergence conditions for ascent methods II: some corrections. *SIAM Review 13, p. 185-188.*

Wurbs, R.A. 1996. *Modelling and Analysis of Reservoir Systems Operation.* Prentice Hall Inc., Englewood Cliffs, N.J.

Yeh, W.W-G. 1985. Reservoir management and operations models: a state-of-the art review. *Water Resources Research, 21(12), p. 1797-1818.*

Young, G.K. 1967. Finding reservoir operation rules. *Journal of Hydraulic Engineering, ASCE, 91(HY6), p. 297-321.*

Yung-Hsin Sun, W-G. Yeh, and Nien-Sheng Hsu. 1995. Generalized network algorithm for water-supply-system optimization. *Journal of Water Resources Planning and Management, 121 (5) September/October.*

Zadeh, N. 1973. More pathological examples for network flow problems. *Mathematical Programming, 5, p. 217-224.*