INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0500
NOTE TO USERS

This reproduction is the best copy available

UMI
STORAGE AND PUMPING SYSTEM OPTIMIZATION USING GENETIC ALGORITHMS

BY

TOBIAS A. BLOCH

A THESIS SUBMITTED IN CONFORMITY WITH THE REQUIREMENTS FOR THE DEGREE OF MASTER OF APPLIED SCIENCE GRADUATE DEPARTMENT OF CIVIL ENGINEERING UNIVERSITY OF TORONTO

© Copyright by Tobias A. Bloch 1998
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

0-612-40917-1
ABSTRACT


Water distributions systems generally do not operate at or their optimal points. The operational procedures of these systems should be reworked to make them more economically efficient. The ability to optimize (i.e., for economic efficiency) the operation of water distribution systems is hindered by the dimensionally (i.e., many different subsystems to optimize simultaneously) inherent in these systems. This is especially true for small time steps. pumping systems utilizing multiple pumps and distribution networks with multiple storage tanks.

A branch of computer science called Genetic Algorithms has been shown to be very effective for optimizing complex systems. Genetic Algorithms are utilized in Two computer programs developed in this thesis. These programs were tested against simple water distribution systems and produced optimal solutions in a very short period of time. The architecture (i.e., program structure) for future computer programs that are based on the platform created in this thesis is also presented. These proposed versions should be useful in the optimization of large water distribution systems, something which has never been possible before.
ACKNOWLEDGMENTS

I wish to express my gratitude to my supervisor, Professor Bryan W. Karney, for his continuous guidance, direction and patience in the writing of this thesis. I would also like to thank my second reader, Professor Robert C. Andrews for his help in the editing of this thesis. I would like to thank the Natural Science and Engineering Research Council of Canada for their financial support of my studies.

I would like to thank the graduate students of the Environmental Engineering Section of the Civil Engineering Department for a very enjoyable and entertaining two years. Especially Martin Pendlebury with whom I shared an office.

Finally, my family, whose support for all of this made it all possible.

Tobias A. Bloch
# TABLE OF CONTENTS

1. Introduction ........................................................................................................... 1

2. The System and its Costs  
   2.1 The Water Distribution System Structure ...................................................... 4  
   2.2 Water Demand Rates ...................................................................................... 6  
   2.3 The Control Volume and System Constraints ................................................ 8  
   2.4 The Cost of the Pumping System .................................................................. 9  
   2.5 Cost of the Elevated Storage Tanks ............................................................... 10  
   2.6 Cost of Electricity ............................................................................................ 11  
   2.7 Discount rate, Inflation rate and Operating Life ............................................ 12  
   2.8 Summary ........................................................................................................ 15

3. The Fundamentals of Genetic Algorithms  
   3.1 Introduction .................................................................................................... 17  
   3.2 Natural Evolution: The Inspiration ................................................................. 17  
   3.3 Genetic Algorithms ......................................................................................... 19  
   3.4 Crossover ......................................................................................................... 22  
   3.5 Crossover Methods Comparison .................................................................... 26  
      3.5.1 One-point crossover ............................................................................... 30  
      3.5.2 Two-point crossover .............................................................................. 30  
      3.5.3 Uniform crossover ................................................................................... 31  
   3.6 Parent Selection ............................................................................................... 31  
      3.6.1 The Roulette Wheel Technique ............................................................... 33  
   3.7 Evaluation and Fitness Value .......................................................................... 36  
   3.8 Bit Mutation ..................................................................................................... 37  
   3.9 Summary ........................................................................................................ 41

4. The Structure of GAHOS  
   4.1 Introduction .................................................................................................... 43  
   4.2 Overview of GAHOS-1 .................................................................................. 43  
   4.3 Overview of GAHOS-2 .................................................................................. 45  
   4.4 Evaluation ....................................................................................................... 47  
   4.5 Reproduction ................................................................................................... 51  
   4.6 Mutation .......................................................................................................... 54  
   4.7 Input and Output .............................................................................................. 57  
   4.8 Model Validation ............................................................................................. 59  
      4.8.1 First Stage ............................................................................................... 60  
      4.8.2 Second Stage ......................................................................................... 61  
      4.8.3 Third Stage ............................................................................................ 62  
   4.9 Summary ........................................................................................................ 65
5. Future Versions

5.1 Introduction.................................................................................. 67
5.2 GAHOS-3 Blueprints................................................................. 67
5.3 GAHOS-4 Blueprints................................................................. 70

6. Conclusions and Recommendations........................................... 73

References.......................................................................................... 75

Appendices

Appendix 1 Results of the effect of S/C ratio on Daily Electricity Cost ....... 78
Appendix 2 Results of the effect of magnification exponent on Daily Electricity Cost ................................................................. 83
Appendix 3 Generating an equation for the capital cost of storage reservoirs ................... 85
Appendix 4 Affect of the mutation values on the $G_p$ value .................... 87
Appendix 5 Tables concerning the model validation (third stage) ................. 91
Appendix 6 Average day water demand rates (AVRDAY 1) ..................... 93
Appendix 7 Description of various function used in the GAHOS computer code ................................................................. 94
Appendix 8 GAHOS-1 computer code ............................................. 96
Appendix 9 GAHOS-2 computer code ............................................. 111
Appendix 10 The Roulette wheel subroutine ....................................... 134
Appendix 11 Starting storage to capacity ratio ..................................... 147
Appendix 12 ENUMERO computer code ........................................... 153
Appendix 13 Toronto Hydro Electricity Rates ....................................... 170
Appendix 14 Input file bit-15.txt .................................................... 174
Appendix 15 Input file test-15.txt ................................................... 176
Appendix 16 Output file showing the effect of the absolute-best function .... 178
LIST OF FIGURES

2.0 The water distribution system layout ........................................... 5
3.0 GAHOS Flow Chart ........................................................................ 21
3.1 An example of a one-point crossover ............................................. 23
3.2 Example of a one-point crossover using “Head” and “Tail” illustrations ... 24
3.3 An example of a two-point crossover ............................................. 25
3.4 An example of a three-point crossover ......................................... 25
3.5 An example of the uniform crossover method .................................. 26
3.6a An example of the schema principal in a real engineering problem ...... 27
3.6b The solution to the real engineering problem .................................. 28
3.7 Determining all the schemata for a particular chromosome .............. 29
3.8 Problematic one-point crossover ................................................. 30
3.9 The roulette wheel parent selection algorithms .............................. 34
3.10 An example of the roulette wheel parent selection method .............. 34
4.0 Chromosome structure of GAHOS-1 ........................................... 45
4.1 Chromosome structure of GAHOS-2 ........................................... 47
4.2 The effect of the absolute-best function ....................................... 53
4.3 The Bit Switching .......................................................................... 56
4.4 An example of binary addition ....................................................... 61
5.0 GAHOS-3 pumping system ............................................................. 66

LIST OF TABLES

3.0 The common method of bit mutation ............................................. 38
3.1 ICMM vs. CMM run times .............................................................. 40
4.0 A summary of the three versions of GAHOS used in the validation .... 60
4.1 Cost properties for five Storage tanks of varying sizes .................... 62
4.2 Cost properties for five different pumps ......................................... 63
4.3 Total System Cost for all possible combinations ............................... 63
List of Abbreviations and Acronyms

B  Number of bit mutations per one hundred bits
C  The maximum capacity of the storage tank
C_E  Daily electricity cost
C_M  Common Mutation Methods
C_P  Capital cost of pump
C_S  Capital cost of storage tank
C_V  Control Volume
G_A  Genetic Algorithm
G_A-HOS-1  First version of GAHOS
G_A-HOS-1  Second version of GAHOS
G_A-HOS-3  Third version of GAHOS
G_A-HOS-4  Fourth version of GAHOS
G_A-S  Genetic Algorithms
G_P  The number of generations before optimal solution reached
H  Head of water
I  Inflow Rate
I_C_M  Individual Chromosome Mutation Method
L  The length of each chromosome in terms of number of bits
M  Magnification factor
M  The number of bits to mutate per chromosome chosen for mutation
N_S  Network Solver
O  Outflow Rate
O&M  Operation and Maintenance
P  Magnification exponent
P  The mutation probability per chromosome
Q  Water flow
R_P  O & M costs of the pump as a percentage of C_p
R_S  O & M costs of the storage tank as a percentage of C_s
R_W M  The Roulette Wheel Method
S_C_R  Starting water volume to maximum Capacity volume Ratio
T_D H  Total Dynamic Head
V_f  Volume of water in the CV
1. INTRODUCTION

Nearly 7 percent of the electricity produced in the United States is consumed by municipal water utilities (Jamieson at el., 1989). The vast majority (typically 95%) of this electricity cost is associated with running the pumping system. These figures are based on data collected in the United States, however they are comparable to those of Canadian municipal water utilities. The vast majority of these water utilities do not operate at or near their optimal levels (Ormsbee and Walski, 1989). There is much room for improving the operational and design aspects of municipal water utilities (Ormsbee and Walski, 1989).

There are several methods by which water supply operations can be optimized. The two most popular methods are Dynamic programming and Lagrangean relaxation (beckwith and Wong, 1995). However, both these methods require significant computer resources and are inadequate for real time optimization. Dynamic programming suffers from explosion in dimensionally and thus finds it difficult to solve (i.e., optimize) complicated water distribution systems (beckwith and Wong, 1995). Lagrangean relaxation find complicated systems difficult as well since it can become entrapped in local minima and thus never find the true minimum cost solution (i.e., the optimal solution).

In this thesis an optimization tool called Genetic Algorithms (GAs) was used to optimize the operations and design of a hypothetical water distribution system (referred to as the “system” or “problem”). Genetic Algorithms were inspired by the theory of
nature selection. Using GAs has been shown to be a very effective approach for optimization purposes (beckwith and Wong, 1995). The research conducted during this thesis produced two computer programs (i.e., two versions), both these programs utilize the principals of GAs. The programs were entitled under the heading of GAHOS (Genetic Algorithm Hydraulic Optimization Software). In this thesis “GAHOS” is used to refer to the software (i.e., all versions).

The first version of the program was called GAHOS-1 and was created as a tool for the optimization of the day to day operation of a water distribution system. GAHOS-1 is capable of determining the optimal pumping schedule as well as the optimal way of managing the storage reservoir. A pumping schedule is the list of times the pump is on or off. Electricity costs vary between different times of a day. Reducing the amount of hours the pumping system consumes electricity during the expensive period is an effective way of decreasing the operational costs of the water utility.

GAHOS-2 was created as a tool for the design (and operation) of efficient water distribution system. This version of the software can also optimize the pumping schedule and storage reservoir management in the same manner as GAHOS-1. However, GAHOS-2 goes further by determining the best pump and storage tank type to install. When operating GAHOS-2, the user inputs an inventory of pumps and storage tanks that the program can consider in its optimization procedure. From this inventory, GAHOS-2 will determine the most optimal combination of pump, storage tank, pumping schedule and storage reservoir management procedures.
Chapter 2 describes the creation of the hypothetical water distribution system, as well as how this hypothetical system differs from one that can be found in practice. Water demand rates and system constraints are discussed. Capital cost equations for the purchase of pumps and reservoirs as well as operation and maintenance liabilities and electricity costs are presented. The Chapter finishes by introducing the equations that are used to evaluate the performance of different solutions (i.e., combinations of pumping schedules, pump types, storage types etc.).

The reader is presented with a brief history and basic principals of Genetic Algorithms in chapter 3. The reader is introduced to a novel mutation method developed in this thesis. This method is different than all other documented methods and is many times quicker (i.e., computationally more efficient) than other methods. The chapter also discusses mutation methods, crossover methods and parent selection methods.

Chapter 4 describes the subroutines used in GAHOS as well as the typical parameter ranges calibrated for the input data presented in the appendices. The validation procedure that GAHOS-1 and GAHOS-2 underwent are also documented in chapter 4. Chapter 5 which provides recommendations for future computer programs based on the GAHOS platform. These recommendations will enhance the engineering usefulness of GAHOS. The thesis finishes with a summary of the conclusions and recommendations.
2. THE SYSTEM AND ITS COSTS

2.1 The Distribution Network Structure

The first task in any optimization project regardless of its numerical method (i.e., Dynamic programming, Genetic Algorithms, Lagrangean relaxation, etc.) is to define the system's structure. This promotes a better understanding the system's internal components and their purpose. In this thesis, the author has constructed a hypothetical water distribution system with the intention of partially optimizing it. The term partially optimizing is used since only a few components of the system will be optimized (the pumping device and schedule, as well as the size of the reservoir).

The physical structure of the hypothetical distribution system is for the most part similar to those employed in practice. However some simplifications and assumptions were necessary to cope with the complex and largely uninvestigated nature of the problem. This approach was prudent since in the beginning of this research it was not known how well genetic algorithms would cope with a multi level optimization.

The water distribution system consists of four major components, a wet well, pumping system, reservoir and distribution network. The wet well stores the potable water that has been processed through the water treatment plant. The amount of water retained by the wet well is of sufficient quantity so as to ensure an ample supply of water to the pumping system, even during times of high water demand rates. The wet well acts
as a buffering mechanisms, aiding the water treatment plant combat fluctuating water demand rates.

**Fig. 2.0: The water distribution system layout**

A typical pumping system consists of pump(s) and valve(s). The number of possible configurations of pump(s) and valves is limitless (each version of the software plainly defines the specific configuration used in its pumping system). Water is transported from the wet well through a pumping system and up into an elevated water reservoir. From the reservoir, water flows by gravity into the distribution network. The major simplifications that make this system different from one that may be encountered in practice are listed below:

1. The pumping system consist of only one pump (with no bypass valve), which is only allowed to operate at one speed. This means that the pump can be represented by only three parameters; its head, discharge and efficiency. Whatever the pump speed chosen, the corresponding head must be large enough to allow the water to reach the
top of the reservoir (the pressure head provided by the pump must be equal to or greater than the elevation difference between the pumping system and the top of the storage tank).

2. All water that enters the distribution network must first pass through the pumping system and reservoir. The pumping system can not directly supply water and pressure to the distribution network.

3. The energy required to supply water to the nodes at above a specified minimum pressure (i.e. 30m of head) will be provided by the elevation difference between the reservoir and the distribution network. This assumption holds true for all water demand rates.

Although these assumptions are constrictive and make the initial versions of the GAHOS impractical (GAHOS-1 and GAHOS-2), they were initially necessary. The knowledge and data gained by experimenting with this simplified water distribution system was extremely beneficial. The experiments conducted by applying GAHOS-1 and GAHOS-2 have enabled the author to begin preliminary design of future versions of the software that avoid these assumptions (GAHOS-3 and GAHOS-4 which are discussed in chapter 7).

2.2 Water Demand Rates

The water demand rate is defined as the sum total of all the water drawn from all the nodes in the distribution network during a specific block of time. In this thesis, these rates are given in cubic meters per second and are partitioned into one hour sections. The
24 water demand rates representing the 24 hours in a day are defined in all GAHOS input files. These rates are located in the second block of input data, following the electricity cost parameters (see Appendix 14 and 15 for an example of an input file).

Two daily flow demands are of interest: i) maximum day and ii) average day. The maximum day demand rates represent the critical demand requirement the system might encounter. The maximum day demands are often used to test how the system functions during these extreme periods (Walski, 1993). The mean rate of water consumption during the maximum day is an important value. It represents the threshold with respect to pump flow. Any pump that can supply water at a rate higher than or equal to the mean flow rate is eligible for consideration as a possible option. However, all pumps that supply water at a rate lower than the mean flow rate can never be considered. This is because these pumps can never replace the water drawn from the reservoir within 24 hours of the start of the maximum day simulation.

The average day flow rates represent the average (from historical data) demands that the system may encounter. The average day rates are lower than the maximum day rates, yet they are important still. The average day simulation provides a more realistic estimation of how the system will function during day to day operations (Ormsbee and Walski, 1989). It is for this very reason that in GAHOS, the annual electricity cost is estimated by taking the electricity cost of the average day simulation and multiplying it by 365. In both GAHOS-1 and GAHOS-2 the average day simulation provides the basis for estimating the electricity cost associated with the pumping operations. The average
day water demand rates used in this thesis are called AVRDAY-1, and are included in Appendix 6.

2.3 The Control Volume and System Constraints

A water distribution system must balance the supply and demand of water and its pressure. Previously it was assumed that the pressure requirements of the network were met automatically by the elevation head of the storage tank (assumptions 2 and 3 located in section 2.1). Thus the problem (or operators obligation) is now reduced to the satisfaction of the water demands generated by the consumers (i.e., ensure that the storage reservoir never runs out of water).

The best way of tracking the quantity of water in the storage reservoir is using what is called a Control Volume (CV). Since the storage tank is the only link between the pump flow (i.e., inflow) and the network demand (i.e., outflow) it must be included within this CV. Because the pump can not directly introduce water into the distribution network (all water must pass through the storage tank), than the CV can simply be defined as the storage tank itself.

The CV has only one inflow and outflow point. The inflow rate (I), is equal to the pump displacement when the pump is turned on. When the pump is turned off, I is equal to zero. The outflow rate (O), is equal to the water demand rate. Assuming incompressible flow the control volume equation is given as

\[
\frac{dV_r}{dt} = I - O
\]  

(2.0)
in which $V_f$ is the volume of water on the CV (storage tank). The above equation is used to track the amount of water in storage (Miles and Moore, 1995).

There are two system constraints that must be satisfied. The first is that the demand flow (generated by the average day demand data) must at all times be met. The second constraint is that $V_f$ must be greater than zero and less than the capacity of the storage tank. Equation 2.0, provides the means by which these constraints are related.

2.4 Cost of the Pumping system

The costs associated with a pumping system can be divided into capital, O&M (operation and maintenance) and electricity costs. This section discusses the first two, while the third group (electricity cost) will be covered in the section 2.6 entitled “Cost of Electricity”. Capital cost of a pump is the sum of the purchase price plus any shipping and installation costs. While O&M cost is the cost generated by maintenance, repair, wages of operators etc.

There are hundreds if not thousands of pump manufactures around the world that produce centrifugal pumps for water-supply applications. With each manufacturer producing several different pump sizes the results in an almost overwhelming number of pumps (that can be considered in the pump selection process).

Each manufacturer prints catalogs that describe the pumps and provide important characteristics and specification (Walski, 1993). These include: purchase price,
efficiency vs. discharge graph, discharge vs. head graph, cost of replacement parts and estimated life of various internal components.

Since the purpose of this project was to simply create and validate GAHOS, it was not vital to use genuine pump data (i.e., discharge, head and efficiency), only data that is reasonably accurate. Thus a hypothetical inventory of pumps and their properties was constructed (these pumps exhibit realistic properties) (Clingenpeel, 1983). The capital cost of these pumps was approximated using equation 2.1,

\[
\text{Capital cost} = \$ \ (690 \ 000) \ Q^{0.7} H^{0.4} 
\] (2.1)

\[Q = \text{water flow} \ [\text{m}^3/\text{s}]\]
\[H = \text{head of water} \ [\text{m}]\]

this expression was adopted from Walski et al. (1987). Equation 2.1, was used to approximate the capital cost of the pumps in the inventory. The yearly O&M cost of these pumps is given as a percentage of the capital cost which in this study varied between 5% and 15% (the more expensive high efficiency pumps were around 5%, while the cheaper lower efficiency pumps were closer to 15%). This percentage value can easily be altered and can be gradually increased by the use of an inflationary constant (which is discussed in section 2.7).

2.5 Cost of the Elevated Storage Tanks

A hypothetical inventory of elevated storage tanks similar to the pump inventory was also constructed. The O&M costs were assumed to be a percentage of the capital cost and the user may alter them via the same inflationary value (see section 2.7) as the
pumps. The capital cost of elevated storage tanks was approximated using the following equation,

\[
\text{Capital cost} = $300\,000 + 140\,C
\]

\(C = \text{the maximum capacity of the tank [m}^3]\)

equation 2.2 was created via a linear regression routine (Appendix 3) from data presented in Muir (1991).

2.6 Cost of Electricity

There are two methods by which electric utilities bill their clients. The first is time-of-use rate structure. The second is the peak-demand (or peak consumption) rate structure (Lackowitz and Petretti, 1983). The time-of-use rate structure is a proactive method by which utilities can penalize user that demand power during peak periods. The electric utility charges a higher price (per kilowatt-hour) for electricity during peak-period and a lower price at off-peak hours (Muir, 1991). Utilities typically define peak periods as 09:00 to 21:00, however this is purely dependent on the electricity consumption pattern of the population. Toronto Hydro charges its customers more in winter than it does in summer (when comparing winter peak period rates to summer peak period rates, and doing the same for the off-peak comparison) (Appendix 13). Regardless the season the off-peak period rates are substantially lower (peak periods can be as much as 3 times more expensive than off-peak periods) than the peak period rates (Chao, 1979). For example, Toronto Hydro gives its customers a 58% discount in winter and a 63% discount in summer (Appendix 13).
The peak-demand rate structure is typically defined by the highest demand charge established in any 15-minutes window for the month (Clingenpeel, 1983). This type of rate structure is only applicable to customers with large monthly demands (greater than 2500 kW).

The peak-demand rate structure is most applicable for pumping stations with a battery of pumps. If an unusually large number (more pumps on at one time than is normal) of these pumps are active, then the probability of peak-demand rate billing increases (Walski and Ormsbee, 1989). Since both GAHOS-1 and GAHOS-2 are based on a single pump system. This justifies the use of the time-of-use rate structure (since peak-demand rate structure is only applicable for large electricity consumers using multiple pumps). The future versions of GAHOS (GAHOS-3 and GAHOS-4) proposed in chapter 7 would incorporate many pumps and these proposed versions of the software would utilize both billing methods.

2.7 Discount rate, Inflation rate and Operating life

In order to successfully complete an economic analysis of the hydraulic system we must group all system costs into one economic indicator (which must incorporate all applicable costs as well as the time value of money, inflation and operating life). The most commonly used indicator is referred to as present worth or its opposite present cost (White et al. 1989). Present day cost is used as an indicator since it is more pleasing to use positive values to represent costs, then negative values representing benefits.
In sections 2.5 and 2.6 the capital cost of both pumps and storage tanks was defined. For the sake of simplification, the operator of the hydraulic system is assumed to pay the capital cost of the pump and reservoir tank on the first day of the project life. These capital costs by definition are already discounted to the present day and such need not be manipulated. There are cases in which the operator does not possess the capital to pay for the pump and storage tank entirely. Under such circumstances the operator typically borrows the capital in the form of a bond or even a loan. In these cases the rate of interest, amount borrowed and time for debt repayment must be used to adjust the present worth of the capital cost of the pump and storage tank. However, the overall changes to GAHOS are minimal and would total less than 5 lines of code.

There are also two other costs that must be accounted for. The first is the daily electricity cost, and the second is the O&M costs of the pump and storage tank. Both the electricity cost and O&M costs are assumed to by paid at the end of each of every operating year throughout the life of the project (or time horizon). Since the life of a project can be anywhere from 10 to 40 years, the time value of money as well as the potential for inflation must be considered.

For simplicity, it is assumed that the rate of inflation of both electricity and O&M cost are subject to the same rate of inflation of r %. Thus, by multiplying the first year’s electricity and O&M costs by a present worth factor and adding to that the capital costs of the pump and storage tank, all the costs of the entire project are summed into a single present worth total cost. The present worth factor is defined as:
\[
(P/A \ i \ .r \ .n) = \left[ \frac{\left(\frac{1+i}{1+r}\right)^n - 1}{(i-r)\left(\frac{1+i}{1+r}\right)^n} \right]^{(r)}
\]

\[(P/A \ i \ .r \ .n) = \text{pronounced P given A or present worth given an annuity of n years, at a discount rate of i, and an inflation rate of r.}\]

Equation 2.3 was taken from White et al. (1989).

Using the present worth factor we can sum all the costs of the hydraulic system:

\[
\text{Total Project Cost} = C_P + C_S + (P/A \ i \ .r \ .n) \cdot (365 \cdot C_E + C_P \cdot R_P + C_S \cdot R_S)
\]

\[
C_P = \text{capital cost of pump [\$]}
\]

\[
C_S = \text{capital cost of storage tank [\$]}
\]

\[
C_E = \text{daily electricity cost [\$/day]}
\]

\[
R_P = O \ & \ M \text{ costs of the pump as a percentage of } C_P.
\]

\[
R_S = O \ & \ M \text{ costs of the storage tank as a percentage of } C_S
\]

The daily electricity costs (\(C_E\)) is calculated from equations 2.5 (from Roberson and Crowe, 1993) and 2.6 shown below;

\[
P = \frac{QH\gamma}{\eta}
\]

where \(P\) is the power requirement [\(kW\)], \(\eta\) is the pump efficiency, \(Q\) is the discharge rate [\(m^3/s\)], \(H\) is the pump head [\(m\)] and \(\gamma\) is the specific weight of water [\(kN/m^3\)]. In order to convert the power value calculated in equation 2.5, we must first manipulate it using the electricity rate as shown below;

\[
C_E = P \cdot (\text{Energy Charge}) \cdot (\text{Time Period})
\]
where the Energy Charge is dictated by the electrical utility [$/kW-hr] and the Time Period is the length of time the pump was in use.

2.8 Summary

This chapter focused on two major areas: i) the physical organization of the water distribution system, and ii) the economic attributes of that same system. The topics discussed in this chapter are essential in order to understand the design and operation of GAHOS. A list of simplifications (i.e., assumptions) is presented below summarizing those simplifications presented in this chapter. The list is applicable to both GAHOS-1 and GAHOS-2.

- The pumping system consists of a single pump, which operates at one speed.
- Any pump can be represented by only three parameters; head, discharge and efficiency.
- All water that enters the distribution system must pass through the pumping system and storage reservoir.
- The elevation head of the storage reservoir provides all the pressure needed to satisfy all the requirements of the distribution network.
- Water demand rates are presented as the sum total of all the demand rates (form all the nodes) of the entire distribution network.
- The water level (in terms of volume) in the storage reservoir must be maintained between zero and maximum capacity.
- The economic analysis conducted within the software is based on the average day water demand rates.
NOTE TO USERS

Page(s) not included in the original manuscript are unavailable from the author or university. The manuscript was microfilmed as received.

16

This reproduction is the best copy available.

UMI
3 THE FUNDAMENTALS OF GENETIC ALGORITHMS

3.1 Introduction

In the previous chapter the reader was introduced to the physical and economic attributes of the hypothetical water distribution system developed in this thesis. As discussed before, this water distribution system would be partially optimized using a computer program called GAHOS. The 'brain' (or numerical engine) that runs GAHOS belongs to a subset of computer science called Genetic Algorithms.

Genetic Algorithms (or GAs) are based on principals developed by scientist researching the theory of nature evolution. Genetic Algorithms when used in system optimization have shown great promise (Miles and Morre, 1995). The abilities of GAs were reconfirmed by the experiments conducted on two versions of GAHOS (see section 4.7). The purpose of this chapter is to give the reader the fundamental knowledge required to understand how genetic algorithms function.

3.2 Natural Evolution: The Inspiration

Looking at the complexity and diversity of biological species on our planet, one is left with curiosity and admiration for "whom" ever or "what" ever has made this possible. For the last two centuries biologists have invested much of their time (Back, 1996), determining the principals behind the "what". In the course of their research they have given their field a name - natural evolution (Back, 1996).
Even as we approach end of the 20\textsuperscript{th} century, the mechanisms of natural evolution are not fully understood. However some general attributes of the theory are widely accepted (from Davis 1991):

- "Evolution is a process that operates on chromosomes rather than on the living beings they encode."

- "Natural selection is the link between chromosomes and the performance of their decoded structures. Processes of natural selection cause those chromosomes that encode successful structures to reproduce more often than those that do not."

- "The process of reproduction is the point at which evolution takes place. Mutations may cause the chromosomes of biological children to be different from those of their biological parents, and recombination processes may create different chromosomes in the children by combining material from the chromosomes of two parents."

- "Biological evolution has no memory. Whatever it knows about producing individuals that will function well in their environment is contained in the gene pool - the set of chromosomes carried by the current individuals - and in the structure of the chromosome decoders."

Through the late 1960s and early 1970s, John Holland (working at the University of Michigan) conducted research into merging the principals of natural evolution into a...
computer algorithm (Michalewicz, 1992). Holland believed that, by incorporating these principals in a computer algorithm, they might yield a technique which is able to solve very difficult problems in the same way nature does - through evolution. Instead of using DNA to encode individuals, he used strings of binary digits (1's and 0's); he called these binary strings *chromosomes*. Holland (1975) observed that a population of low-grade solutions which undergo simulated evolution could yield advanced solutions to difficult problems. As the potential of these types of algorithms became more evident, Holland named them *genetic algorithms*.

3.3 Genetic Algorithms

System optimization can be described as the search for the best solution. Solutions can be compared to one another by way of their quality. In Genetic Algorithms this quality is called "fitness". The "fitness" value assigned to a solution reflects its ability to meet the objective(s) and constraint(s) associated with the system (Halhal et al., 1995). The better the solution, the higher is its fitness value.

A solution's chromosome can be thought of as the detailed bill of materials. The chromosome contains all the information required to perfectly depict the particularities of the solution. In order to simplify the labor required in the evolution of these chromosomes, their contents are encoded in a structure much like the DNA double helix. However, this results in the inability to directly acquire the fitness of the encoded chromosome. One can parallel this with trying to analyze a chromosome that encodes a
particular combination of a certain type of pump, storage tank size and pumping schedule without testing (or simulating) this combination first.

In order to acquire the fitness value, a chromosome must pass through two different procedures (a "decoder" and an "evaluator"). In the "decoder" the data stored within the chromosome is altered into a rational set of instructions. In the "evaluator" the data composed in the decoder is evaluated and is assigned a fitness value.

Genetic Algorithms just like species in nature require a number of breeding individuals in order to guaranty the survival of the species. The sum of these breeding individuals is called a "population". The size of a population used in GAs varies with respect to the type of system involved. A normal range would be between 10 and 1000 individual chromosomes. The literature reviewed in this thesis recommends that the population size remain constant through the entire simulation (i.e., generation after generation). This is exactly the case in both GAHOS-1 and GAHOS-2 (as well as the proposed GAHOS versions presented in chapter 5).

Fig. 3.0, shown on the next page is a flowchart of GAHOS. The figure illustrates the execution order of the main subroutines. Before reviewing Fig. 3.0, it is advantageous to define the purpose of these subroutines in simple terms:

**Crossover** - The creation of new chromosomes by mating current chromosomes (referred to as parent chromosomes), after which the parent chromosomes are deleted.
Fig. 3.0: GAHOS Flow Chart

Start

Input system data

Process electricity and demand data

Initialize population of chromosomes

More generation required?

True

Crossover

Mutation

Evaluation

Roulette Wheel

False

Output simulation data

Return to screen the best chromosome

End
Mutation - Randomly applying mutation to a fixed percentage of the new chromosome population.

Evaluation - Evaluate each chromosome in the population and assigning it a fitness value.

Roulette Wheel - The roulette wheel method is a specific technique that is classified as parent selection. This subroutine chooses which chromosomes of the current generation would be fit enough to be re-labeled as parent chromosomes.

The remainder of this chapter is devoted to a discussion of the various subroutines depicted in Fig. 3.0. The material presented in the remainder of this chapter is essential in creating for the reader, a better understanding of Genetic Algorithms. Also, this chapter introduces some new Genetic Algorithm techniques developed during the completion of this thesis.

The review of the crossover techniques described in section 3.4 were based upon a review of these publications: Back (1996), Davis (1991), Michalewics (1992) and Syswerda (1989).

3.4 Crossover

Crossover occurs when two parents exchange non-overlapping parts of their respective chromosomes to produce children. The order of the crossover is defined by the number of sections which the parent's chromosome are divided into. Below is an example of a one-point crossover of a 6 character chromosome.
The chromosome used in Fig. 3.1 used 6 characters (or sectors), where each sector can be occupied by either a zero or a one. This type of encoding (i.e., using zeros and ones) is often used for systems that require scheduling of simple devices. The pump as defined in this thesis (i.e., GAHOS-1 and GAHOS-2) is considered simple since it can be either on or off (i.e., only two choices). The digit zero is used to represent the pump being turned off, while the digit one represents the pump being turned on. In order to simplify Fig. 3.1, it was given a chromosome of only 6 sectors (schedule made of 6 time units). The simulations done in GAHOS are always 24 hours long and thus the smallest pump scheduling chromosome is 24 sectors long (using 1 hour time steps). The largest pump scheduling chromosome is 96 sectors long (using 15 min. time steps).

**Fig. 3.1:** An example of a one-point crossover.

| Two parent chromosomes (1 and 2) are to undergo a one-point crossover after the fourth character position (i.e. called loci). Parent 1: (1,1,1,1,1,1) and Parent 2: (0,0,0,0,0,0). Parent 1: 1111 | Parent 2: 0000  |
| Child 1: 1111 | Child 2: 0000 |
| 00              | 11              |

The loci at which the crossover occurs is randomly selected by the computer. If the computer has returned the integer 2 as the crossover loci, the genetic exchange would have looked like this:

| Parent 1: 11 | 1111 | Child 1: 11 | 0000 |
| Parent 2: 00 | 0000 | Child 2: 00 | 1111 |

One can imagine that during a one-point crossover, each of the two parent chromosomes are divided into two separate sections. The left side (or start) of the chromosome is called the "Head", while the right side (or end) of the chromosome is called the "Tail". This is illustrated in Fig. 3.2, shown below.
Fig. 3.2: An example of a one-point crossover using “Head” and “Tail” illustrations.

<table>
<thead>
<tr>
<th>Parent 1: H1-T1</th>
<th>Child 1: H1-T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent 2: H2-T2</td>
<td>Child 2: H2-T1</td>
</tr>
</tbody>
</table>

Once the parent chromosomes are divided into a “Head” and “Tail” sections, the computer produces two children which use the “Head” of one parent and the “Tail” of the other parent. H1 must be the same length as H2 and T1 must be the same length as T2. However, the “head” and “Tail” sections need not be the same length. As can be observed from Fig. 3.2, a one-point crossover produces two distinct children. The first child (i.e., H1-T2) uses the head of parent #1 and the tail of parent #2. The second child (i.e., H2-T1) uses the head of parent #2 and the tail of Parent #1.

The one-point crossover method is the technique used in both GAHOS-1 and GAHOS-2. This particular techniques was chosen after a careful review of two other methods (a summary of this review can be found in section 3.5). An introduction to the two other crossover techniques is presented below (these are the two-point crossover and the uniform crossover).

A two-point crossover simply requires the definition of a middle section represented as M. The first parent will be represented by H1-M1-T1, and the second parent by H2-M2-T2.
**Fig. 3.3:** An example of a two-point crossover.

<table>
<thead>
<tr>
<th>Parent 1: H1-M1-T1</th>
<th>Child 1: H1-M2-T1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent 2: H2-M2-T2</td>
<td>Child 2: H2-M1-T2</td>
</tr>
</tbody>
</table>

The template used for the two-point crossover technique can be used for higher order crossovers as well. This is accomplished by breaking up the “Middle” part of the parent chromosome into sections. In *Fig. 3.4* shown below, the letter and subscript are used to identify the section, while the number in normal font is used to identify the parent which it belonged to.

**Fig. 3.4:** An example of a three-point crossover.

<table>
<thead>
<tr>
<th>Parent 1: H1-M1-M21-T1</th>
<th>Child 1: H1-M12-M21-T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent 2: H2-M12-M22-T2</td>
<td>Child 2: H2-M11-M22-T1</td>
</tr>
</tbody>
</table>

The final crossover technique that will be described is the uniform crossover method (Syswerda, 1989). The method is best explained as follows. For each bit position of the two children, the computer randomly draws which parent will contribute its bit to which child (*Fig. 3.5*).

Just as in the previous techniques two parents are selected. The computer randomly generates either a 0 or 1 and repeats this procedure as many times as there are bits in the chromosome. This string of zeros and ones is called the template. Once the template is complete an exact opposite of the template, called the anti-template can be constructed. The template is used to construct child 1, while the anti-template is the basis for child 2 (The bits of Parent 2 are bolded in order to show their final locations).
Fig. 3.5: An example of the uniform crossover method

| Parent 1  | 1 0 0 1 0 1 1 |
| Parent 2  | 0 1 0 1 1 0 1 |
| Template  | 1 1 0 1 0 0 1 |
| Anti-template | 0 0 1 0 1 1 0 |
|            | ↓               |
| Child 1   | 1 0 0 1 1 0 1 |
| Child 2   | 0 1 0 1 0 1 1 |

As can be seen in Fig. 3.5, the anti-template is constructed by taking the template and simply substituting a zero in place of a one and visa versa. In both the template and anti-template string, the number one represents parent 1 and number zero represents parent 2.

3.5 Crossover Methods Comparison

In the previous section the reader was introduced to the most current and popular crossover techniques to date (Michalewics, 1992). The current discussion focuses on the advantages and disadvantages of the one and two-point crossover method, as well as the uniform crossover technique. Crossovers that utilize more than two points will not be discussed since they are generally found to be inferior to the other methods (Michalewics, 1992).

Before comparing different crossover methods it is advantageous to first introduce the reader to the notion of the schema. A schema is a special chromosome (i.e., string of bits) that is made of a combination of 1, 0, and "*". The symbol "*" is interpreted as a
don't care symbol. The bit at which the "*" symbol is found can be changed to either a zero or a one with no effect on the solution (Michalewicz, 1992).

A popular question is, how does a schema fit into the scope genetic algorithms? The answer is that genetic algorithms strive to find solutions to difficult problems optimal (but usually settle for near-optimal). Most engineering problems are such their absolute best solutions are impractical. For example, after analysis of a water distribution system, it was concluded that the best solution is to build a storage reservoir with a capacity of 4398.15 m³. Of course no engineer would insist on building that exact storage tank, but would settle for a similar and commercially available substitute. The fitness (i.e., quality) of this substitute solution would be very close to the absolute best solution and both solutions would be practically identical in physical form. In such cases (these as well as in the problem stated in this thesis) it would be advantageous to create a template (i.e., schema) for near optimal solutions. The following example demonstrates the use of a schema.

Fig 3.6a: An example of the schema principal in a real engineering problem

As a civil engineer you are faced with a storage tank size optimization problem. At your disposal are the following eight tank volumes.

<table>
<thead>
<tr>
<th>1200</th>
<th>1400</th>
<th>1600</th>
<th>1800</th>
<th>2000</th>
<th>2200</th>
<th>2400</th>
<th>2600</th>
</tr>
</thead>
</table>

(all in m³)

A colleague has heard of your problem and has written a very advanced GA routine. The colleague has concluded that the optimal tank volume is 1800 m³. Never-the-less you proceeded and write your own less sophisticated GA program, but could not converge on a single answer. For the past 50 generation your program could not reduce the number of possible solutions below 5. The final 50 generation of solutions produces the following tank volume.

2032 2020 2015 1989 1975

How could the use of a schema have helped you converge earlier?
Looking at the 5 solutions your program has generated, one can see that for each of the five solutions the closest storage tank volume is 2000m$^3$. You as a human being can make the judgment to settle for a commercially available tank size. However, how can you tell the computer to do the same thing.

First convert the five solutions into binary form:

- 2032 $\Rightarrow$ 011111110000
- 2020 $\Rightarrow$ 011111100100
- 2015 $\Rightarrow$ 011111011111
- 1989 $\Rightarrow$ 011111000101
- 1975 $\Rightarrow$ 011110110111

As can be seen from the binary form all five chromosomes, they all have identical first five digits (starting from the right side). In fact the schema 01111******* can represent all five of these solutions.

The schema 01111******* can be used as a rounding off method for binary chromosomes in this particular problem. It is important to remember that the schemata for the eight available tank sizes must be done prior to executing the Genetic Algorithm.

The previous example could have been solved without the aid of a schema. By converting the computer solutions (i.e., 2032m$^3$, 2020m$^3$, 2015m$^3$, 1989m$^3$, 1975m$^3$) to commercially available tank sizes (i.e., 2000m$^3$) the program would have terminated sooner as it converged onto 2000m$^3$ tank volume.

Schemata can be useful not only for the purpose of rounding off solutions. There are many situations in which schema significantly aid in the convergence of a solution. Schema are able to do this by identifying bits (sections in the string) that have little or no effect on the fitness of the chromosome as a whole.

Imagine, for instance, a 12 bit chromosome composed of four independent sections. The first section being 3 bits long encodes parameter A. The second section is
3 bits long and encodes parameter B, and so on for section 3 and 4 for parameter C and D respectively. By using a schema generating subroutine the program may identify which if any of the four are important. This type of information would only be available via a sensitivity analysis, which is a large undertaking.

It can be helpful to think of a schema as a way of representing strings. For example, the schema (*,1) represents the strings (1,1) and (0,1). The schema (1,1,0,1) represents one string only: (1,1,0,1). The schema (*,*,*,*) represents all strings of length 4. It is obvious that every schema matches exactly $2^d$ strings, where $d$ is the number of don't care symbols in the schema. On the other hand, a string of length $b$ is matched by $2^b$ schemata. The following example illustrates these concepts.

**Fig 3.7:** Determining all the schemata for a particular chromosome.

<table>
<thead>
<tr>
<th>Determine how many schemata there are for the chromosome (01101) and show all of them.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since $2^2 = 32$</td>
</tr>
<tr>
<td>(01101)</td>
</tr>
<tr>
<td>(**101)</td>
</tr>
<tr>
<td>(0<em>10</em>)</td>
</tr>
<tr>
<td>(*<em>10</em>)</td>
</tr>
<tr>
<td>(<em>11</em>*)</td>
</tr>
<tr>
<td>(<strong>1</strong>)</td>
</tr>
</tbody>
</table>

In the previous pages we have concentrated on understanding the principals of the schema. This was all done for the purpose of assisting the reader through the next three important sections which use the theory of schema as a tool for comparing different crossover methods.
3.5.1 One-point crossover

The major advantage of the one-point crossover is its simplicity. However, in certain cases (especially in long and multi-parameter chromosomes) one-point crossover schemes are slow to converge. One such example is given below, as can be seen from Fig. 3.8 a one-point crossover is unable to combine all of the high performance (highlighted) bits.

**Fig 3.8:** Problematic one-point crossover.

<table>
<thead>
<tr>
<th>Parent 1</th>
<th>1 1 0 1 0 1 1 1 0 0 1 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent 2</td>
<td>0 0 0 1 0 0 1 0 1 0 0 1 0</td>
</tr>
</tbody>
</table>

Using one-point crossover, the above parents are unable to create the schema shown below,

\[(**1** * 0 0 * 1 * 1 0 *)\]

The reader can see this by looking at the way the high performance bits alternate (i.e., zigzag) between the two parent chromosomes. This zigzag action results in the inability to divide the parent chromosomes into a “Head” and “Tail” section that can be recombined into the schema shown above.

3.5.2 Two-point crossover

A two-point crossover is by far the most popular crossover technique (Davis, 1991). The high performance bits in the chromosomes shown in Fig. 3.8, could have been combined via the two-point crossover. However there are schemata that the two-point crossover can not combine. For example the schema (**1,0,0,1,**1) when divided into the schemata (**1,1,0,0,**1) and (**0,0,1,1,**0) can’t be reassembled using the two-point crossover method.
3.5.3 Uniform crossover

The uniform crossover (Fig. 3.5) method has the distinct ability to combine all combinations of schemata. However, the method is quite 'violent' in the sense that it can cause a great deal of harm to whatever is good in a chromosome. It is thus advantageous to not use this method especially in problems which converge easily (Michalewicz, 1992). This is because problems that converge easily require crossover methods proceed gradually (i.e., gentle).

3.6 Parent Selection

Sections 3.4 and 3.5 discussed various crossover methods by which parent chromosomes pass on their genes to the children chromosomes. However, the techniques of choosing which chromosomes would become parents was not discussed.

A population of individuals (parents) reproduce and thus create a new generation of individuals (children). The new population of children are evaluated on the basis of their fitness (i.e., their ability to contend with their environment). Only the fittest children are allowed to become parents and thus pass on their genes to future generations.

The term “survival of the fittest” is the basis of the previous paragraph. It is been used to describe everything from prey and predator evolution to climbing the corporate ladder. Genetic Algorithms utilize the idea of “survival of the fittest” within the parent selection procedure. When an entire generation of chromosomes (i.e., solutions) pass through the parent selection procedure, these chromosomes undergo a process that is best
described as “survival of the fittest”. In the parent selection subroutine, the chromosomes that best meet the objectives and constraints defined for the system are the ones which have the highest chance of passing on their genes. In nature, high performing individuals tend to live longer and thus procreate more than low performing individuals. However, within the realm of Genetic Algorithms (which exist within a computer), all chromosome live the same length of time. Thus, the only way of exercising the idea of “survival of the fittest” is to allow high performing chromosomes more chances of procreating (which takes place within the Crossover subroutine), and do the opposite to low performing chromosomes. In other words, the genes of the fittest chromosomes will survive (by means of passing the genes to children chromosomes).

Sometimes “Survival of the fittest” can also lead to species or population extinction if taken to the extreme. Imagine for a moment a specie that has evolved into a form which fits perfectly with its environment. whose mutation rate is very low and the gene pool of the species is very small. If the environment should suddenly change, there is a good possibility that the species would become extinct.

There are several documented methods of choosing parent chromosomes in GA. By far the most popular one is called the “roulette wheel” (Michalewics, 1992). The basic principals of the roulette wheel parent selection technique is explained in the next section. A comprehensive discussion of how this method is utilized in GAHOS is given in section 4.2.
3.6.1 The Roulette Wheel Technique

Physically, a typical roulette wheel is a circle divided into equally sized slices, with each slice having an equal probability of being chosen. Imagine for a moment a roulette wheel with \( n \) equal slices. Each of the \( n \) slices being allocated to a specific chromosome (i.e., child). Next all \( n \) chromosomes are evaluated and allocated a fitness value. A new biased roulette wheel is constructed. It has \( n \) slices, with the size of each slice being proportional to its chromosome's fitness. In other words, the ratio of slice area to the entire wheel is equal to the ratio of the chromosome fitness to the sum of the entire population's fitness.

By simulating where a ball will stop in this biased roulette wheel we can choose a new parent (i.e., new generation). This ball throwing is repeated \( n \) times. One can imagine that there is the possibility of producing duplicates within this new generation (i.e., high fitness chromosomes are chosen more than once). This is absolutely normal and should not be inhibited. The simplest way to explain this is to parallel this characteristic of GA to nature. In nature, individuals with better than average fitness will live longer than average lives and thus produce a higher than average number of offspring. However, in the make-believe world of GA, all individuals live the same length of time. Thus, each paring of chromosomes is only allowed two children. Therefore the only way to increase the influence of high performance chromosomes is to allow multiples of themselves to be placed in the pool of future parents.
The basic algorithms of the roulette wheel parent selection technique is presented in Fig. 3.9, shown below (Michalewics, 1992).

**Fig. 3.9:** The roulette wheel parent selection algorithm

1. For each generation evaluate all the \( n \) members based on their fitness.
2. Place all the members of the population in an imaginary queue, the queue need not be in any particular order.
3. For each member in the queue assign a number which is the sum of its fitness and the fitness of every member before it, call this value the "running total". The "running total" of the last member in the queue is assigned the special name of "total fitness".
4. Generate \( x \), a random number between 0 and total fitness.
5. Return the first population member whose corresponding running total is greater than or equal to \( x \).
6. Repeat steps 4 and 5 for as many times as there are members wanted in the for coming generation.
7. Repeat steps 1 through 6 as many times as there are generations required.

As previously discussed, certain high performance chromosomes will be represented various times in the population. Thus it should not come as a surprise that low performance chromosomes not be represented in subsequent generations (i.e., become extinct). The proceeding example demonstrates the multiple-representation and extinction characteristics of the roulette wheel parent selection technique.

**Fig. 3.10:** An example of the roulette wheel parent selection method

Imagine 10 chromosomes with 10 distinct fitness values. These chromosomes will not be mutates nor will they undergo any crossovers. Simply these chromosomes will be used to illustrates how the roulette wheel method is biased towards the higher fitness chromosomes in any given population.
The table below represents the fitness and running totals of the original population.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitness</td>
<td>8</td>
<td>2</td>
<td>17</td>
<td>7</td>
<td>2</td>
<td>12</td>
<td>11</td>
<td>7</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Running Total</td>
<td>8</td>
<td>10</td>
<td>27</td>
<td>34</td>
<td>36</td>
<td>48</td>
<td>59</td>
<td>66</td>
<td>69</td>
<td>76</td>
</tr>
</tbody>
</table>

The total fitness of generation 0 (i.e. original generation) is equal to 76. The next step is to generate 10 random numbers between 0 and 76. And to return the first population member whose corresponding running total is greater than or equal to 76 (i.e. steps 4 and 5 simultaneously).

<table>
<thead>
<tr>
<th>Random Number (x)</th>
<th>64</th>
<th>25</th>
<th>55</th>
<th>9</th>
<th>48</th>
<th>9</th>
<th>12</th>
<th>2</th>
<th>57</th>
<th>66</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome Chosen</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

The table above represents the second generation (the first generation is the original population). One can notice that already some chromosomes are represented twice while others have gone extinct.

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>8</th>
<th>3</th>
<th>7</th>
<th>2</th>
<th>6</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitness</td>
<td>7</td>
<td>17</td>
<td>11</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>17</td>
<td>8</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>Running Total</td>
<td>7</td>
<td>24</td>
<td>35</td>
<td>37</td>
<td>49</td>
<td>51</td>
<td>68</td>
<td>76</td>
<td>87</td>
<td>94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Random Number (x)</th>
<th>59</th>
<th>93</th>
<th>48</th>
<th>56</th>
<th>57</th>
<th>2</th>
<th>52</th>
<th>73</th>
<th>57</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome Chosen</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

The above table represents the third generation. As can be seen the high performance chromosome # 3 is present in 6 out of the 10 possible chromosome types. One must remember that this example is with reference to only parent selection, no crossover or mutation is occurring.

The table shown below summaries how the roulette wheel parent selection method converges on chromosome #3.

<table>
<thead>
<tr>
<th>Generation</th>
<th>Chromosomes Chosen to continue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>2</td>
<td>1 1 1 3 3 10 4 7 1 8</td>
</tr>
<tr>
<td>3</td>
<td>7 8 3 7 7 10 3 4 3 10</td>
</tr>
<tr>
<td>4</td>
<td>7 7 5 7 8 3 7 3 7 3</td>
</tr>
<tr>
<td>5</td>
<td>3 7 7 3 7 3 7 3 3 7</td>
</tr>
</tbody>
</table>
3.7 Evaluation and Fitness Value

Section 3.6 discussed how chromosomes are selected to become parents based on their fitness value. It also discussed how the fitness value is obtained from the evaluator subroutine. All the evaluator subroutines used in GAHOS are based on equations 2.0, 2.3, 2.4, 2.5 and 2.6. Equation 2.4, the total project cost is repeated below in order to assistance the reader.

$$\text{Total Project Cost} = C_P + C_S + \frac{(P/A \ i, \ r, \ n) \cdot (365 \cdot C_E + C_P \cdot R_P + C_S \cdot R_S)}{C_P}$$

- $C_P = \text{capital cost of pump} [\$]$
- $C_S = \text{capital cost of storage tank} [\$]$
- $C_E = \text{daily electricity cost} [\$/\text{day}]$
- $R_P = \text{O & M costs of the pump as a percentage of } C_P$
- $R_S = \text{O & M costs of the storage tank as a percentage of } C_S$

In GAHOS-1, equation 2.4 is simplified since the capital and O&M costs of the pump and storage tank are irrelevant.
The total cost assigned to a chromosome is the sum of the cost given by equation 2.4 plus any associated penalties. The convergence of these costs into a fitness value is explained in detail in section 4.2.2.

3.8 Bit Mutation

There are instances during the creation of the children chromosome whereby certain parts of the chromosome are altered in such a way that they differ from both parents. This naturally occurring event is called mutation. It is a sporadic yet important aspect of evolution (both natural and computer oriented). In cases were certain bits in the entire population are identical, it is the only way by which to expand the gene pool. However, mutation to a certain extent bypasses natural selection, since it may create characters in children that may have been extinct. As discussed previously in section 3.2, natural selection is an excellent optimization tool and thus should not be bypassed without reason. In the author's opinion, mutation is better described as the "art" rather than the "science" of balancing population convergence and diversity.

One of the most common methods of bit mutation is to generate uniformly distributed random numbers (U(0,1)) for all bits, and if the random number is below the mutation probability (i.e., frequency), then the bit is altered (Davis, 1991). The mutation probability is defined as the number of bit mutations per 100 bits. The above example used a mutation probability of 0.15, however in most cases the probability is below 8% (or 0.08). The mutation probability best suited for a particular problem or genetic
algorithm can only be determined by experimentation. The recommended mutation parameters of GAHOS-1 and GAHOS-2 are available in chapter 6.

In Table 3.0, when a random number is below the 15% mutation probability (i.e., 0.15) is it bolded. As well, the bits that are mutated are also bolded.

<table>
<thead>
<tr>
<th>Old Chromosome</th>
<th>Random Numbers</th>
<th>New Chromosome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0 1 1</td>
<td>.146 .506 .430 .699 .365 .333</td>
<td>0 0 1 0 1 1</td>
</tr>
<tr>
<td>1 1 1 0 0 0</td>
<td>.466 .955 .470 .642 .038 .794</td>
<td>1 1 1 0 1 0</td>
</tr>
<tr>
<td>1 0 0 0 1 1</td>
<td>.339 .504 .278 .265 .392 .246</td>
<td>1 0 0 0 1 1</td>
</tr>
<tr>
<td>1 1 0 0 1 1</td>
<td>.174 .165 .343 .168 .846 .024</td>
<td>1 1 0 0 1 0</td>
</tr>
</tbody>
</table>

In the author's opinion the method shown in Table 3.0 (taken from Davis, 1991) is cumbersome and slow. Requesting the computer to generate as many random numbers as there are bits is a huge waste of time. A better way to approach the mutation process is on an individual chromosome level. In this thesis such a technique was developed and is called the Individual Chromosome Mutation Method (ICMM).

The mutation method presented in Table 3.0 is called CMM (Common Mutation Method). CMM relays on only one user defined parameter B, the number of bit mutations per 100 bits. ICMM requires three such variables. The first is L, were L is the length of each chromosome in terms of number of bits. The second is P, were P is the mutation probability per chromosome. The third is N, were N is the number of bits to mutate per chromosome chosen for mutation.

\[ B = L \cdot P \cdot N \] (3.0)
\[ B = \text{Number of bit mutation per 100 bits} \]
\[ L = \text{Length of each chromosome} \]
\[ P = \text{Mutation Probability per chromosome} \]
\[ N = \text{Number of bits to mutate per chromosome chosen for mutation} \]

As can be seen from equation 3.0, the product of \( L \cdot P \cdot N \) is actually equal to \( B \), however the ICMM holds a significant advantage over the CMM.

Both the CMM and ICMM achieve the same bit mutation probabilities, only ICMM can do it significantly faster and allows the user to customize the mutation process to the system involved. For example, compare the two methods on a GA which has a population of 1000 chromosomes of 6 bits each. The user would like the bit mutation probability (i.e., \( B \)) to be 0.05. In this example the CMM must generate 6000 \((1000 \times 6)\) random numbers and compare this random numbers to the number 0.05 an equal number of times.

Lets say the user would rather use the ICMM. The user would like to alter one bit in each chromosome designated for mutation. This means that \( N \) is equal to 1, and \( L \) is equal to 6 (6 bits in a chromosome). Therefore \( P \) is equal to \( 0.00833 \sim 0.008 \) (since \( B/[L \cdot N] \) or \( 0.05/[1 \cdot 6] = 0.00833 \)). In other words 0.8% of the chromosomes should be chosen for mutation, with one bit mutated in each of these. The ICMM would require to generate 16 random numbers and only 8 of these need to be compared to the variable \( P \). A remarkable reduction from 6000 random numbers and 6000 comparisons.
Suppose the user wants two bits to be mutated in each chromosome designated for mutation. Therefore $N$ would be equal to 2, and thus $P$ would be equal to 0.0042. These new parameters further reduce the computer time required. More importantly it shows how the ICMM allows for more flexibility over the common method. Table 3.1. shown below compares the execution time between the common method and ICMM (based upon a population of 1000, 6-bit individuals).

**Table 3.1:** ICMM vs. CMM run times

<table>
<thead>
<tr>
<th>Mutation probability (B)</th>
<th>ICMM (sec)</th>
<th>CMM (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.011</td>
<td>0.16</td>
<td>15.4</td>
</tr>
<tr>
<td>0.50</td>
<td>7.8</td>
<td>15.4</td>
</tr>
</tbody>
</table>

Even at a mutation probability of 50% ($B = 0.50$), the ICMM is still superior to its counterpart. As well we can see that at lower mutation probabilities, the difference in run times are significant. It is for these reasons that ICMM is used in all GAHOS versions presented in this thesis.

The ICMM has one inherent shortcoming when compared to the CMM; it may mutate the same bit(s) twice thereby eliminating or reducing the extent of the mutation. The ICMM randomly select chromosomes from the generation that will undergo mutation, it then randomly select bits (positions) within these chromosomes that will be mutated. As one can imagine there exits the possibility that the same chromosome could be selected more then once, and that in the second mutation procedure, some (if not all) of the mutations done in the first mutation procedure may be erased. The likelihood of such an event is very small. For instance, the example suggested in Table 3.1 used a
population of 1000, $L$ equal to 6, $N$ equal to 2 and $P$ equal to 0.0042. For this particular example the probability that a bit would be mutated twice is $1.96 \times 10^{-6}$ (based on $P^2 \cdot \frac{N}{L^2}$). However, if one wishes to eliminate this possibility it may be done using only a few extra lines of computer code. The user can insert an ‘If-Else block’ immediately proceeding the section that generates the random number (that represents the chromosome to be mutated). Using these new lines of code the computer will be able to reject random numbers that were previously picked.

3.9 Summary

Chapter 3 presented the reader with the fundamental principals and techniques of genetic algorithms. The objectives of this chapter were twofold: i) to provide the reader with a general understanding of how genetic algorithms function ii) to increase the readers understanding of how genetic algorithms are employed this thesis. Below is a summary of the most important themes discussed in this chapter.

- GA where inspired by the theory of natural evaluation.
- The solutions used in GAs are called chromosomes.
- The sum of all the chromosomes is called the population.
- The size of the population is kept constant throughout the simulation.
- Chromosomes are in coded form and must be decoded before being evaluated.
- Chromosomes are evaluated for the purpose of determining their fitness value.
- The fitness value reflects the ability of a solution (i.e., chromosome) to meet the objective(s) and constraint(s) associated with the system.
- Crossover occurs when parents exchange of non-overlapping parts of their respective chromosomes to produce children.
• The crossover technique utilized in GAHOS is the one-point crossover method.
• Parent selection is the process by which certain children chromosome are selected to become parents and thus pass on their respective genes.
• The roulette wheel method is the parent selection process used in GAHOS (discussed in further detail in section 4.4 and Appendix 10).
• Mutation may cause the chromosomes of children to include bits that were not present in either of the parent chromosomes.
• The ICMM (Individual Chromosome Mutation Method) holds a substantial advantage over the CMM (Common Mutation Method).

Even though the ICMM is computationally faster than the CMM, both these methods are used in GAHOS-1 and GAHOS-2. The reasons behind this decision can be found in section 4.5.

The next chapter (chapter 4) will discuss in detail how the different versions of GAHOS are able to optimize the hypothetical water distribution system. This chapter will also include a section that describes how GAHOS-1 and GAHOS-2 were validated.
4. THE STRUCTURE OF GAHOS

4.1 Introduction

This chapter summarizes the various subroutines used in GAHOS-1 and GAHOS-2. The purpose of this chapter is to give the reader a better understanding of how GAHOS-1 and GAHOS-2 are structured as well as how to use these programs. The chapter ends with a review of the verification process which GAHOS-1 and GAHOS-2 underwent.

4.2 Overview of GAHOS-1

As previously discussed there are two versions of GAHOS presented in this thesis, GAHOS-1 and GAHOS-2. The first GAHOS-1, was designed as a tool for the optimization of the day to day operation of a water distribution system. The program is capable of determining the most optimal pumping schedule as well as the optimal way of managing the water within the storage reservoir.

The pumping schedule is simply the timetable of processes occurring within the pumping system. Since the hypothetical water distribution system created in this thesis uses only one pump with only two settings (i.e., on or off). The pumping schedule is essentially a list of times the pump is turned on or off.

The amount of water stored in the reservoir at the start of the simulation (i.e., time zero) has a profound effect on the characteristics of the optimal pumping schedule as well as its associated electricity cost (see Appendix 1). In this thesis a variable has been
defined that reflects the quantity of water at time zero, this variable is called the SCR (Starting volume to Capacity volume Ratio). The volume of water at time zero can be found by multiplying the SCR by the maximum storage capacity of the reservoir. GAHOS-1 can simultaneously determine the optimal pumping schedule and SCR value. Since time zero is defined by the user, thus the SCR value determined by GAHOS-1 is the optimal volume of water at that specified time. Also since the program can control the volume of water in the tank (by i) controlling the pump and ii) since the program knows how much water is drawn out) that is equivalent to managing the volume of water in the storage tank throughout the simulation. In other words GAHOS-1 optimizes the pumping schedule and reservoir operation simultaneously for a 24 hour block of time.

Like all Genetic Algorithms, GAHOS-1 uses chromosomes to encode sets of instructions representing solutions. Each chromosome contains a complete (24hr) pumping schedule and an associated SCR value. GAHOS-1 (as well as GAHOS-2) uses time steps between 15 and 60 minutes (typically either 15, 20, 30, 40, 45 or 60 minute time steps).

Each and every chromosome in GAHOS-1 has at its disposal a 97 sector long array. The first 96 sectors are reserved for the pumping schedule (each sector storing either a zero or one), while the 97th sector is reserved for the SCR value (storing an integer between zero and 1000). A pictorial view of the chromosome structure used in GAHOS-1 is shown below as Fig. 4.0. The entire chromosome is defined as a static array (array of integer values) and thus does not change (static array as opposed to a
dynamic array is used). When the time step is 15 minutes, the entire chromosome is utilized (i.e., sector 1 through 97 are storing relevant instructions). However, when the time step is defined as one hour, only the first 24 and the 97th sectors are utilized, sector numbers 25 through 96 are empty. This of course is a waste of memory, however the simplicity of static arrays is a major advantage.

**Fig. 4.0: Chromosome structure of GAHOS-1**

As mentioned previously the first 96 sectors can either be 0 or 1, where 0 is pump off and 1 represents the pump on (the pumping schedule is 96 sectors long because there are 96 fifteen min. sectors in 24 hours). The 97th sector is a whole number between 0 and 1000 (representing SCR x 1000). An integer requires less computer resources, and thus it is advantageous to define the array as integer, with the fraction created by dividing the integer of the 97th sector by 1000. This would be equivalent to a three decimal float.

### 4.3 Overview of GAHOS-2

GAHOS-2 was created with the intention of becoming a design tool for water distribution systems. This version of the software can also optimize the pumping schedule and SCR value in the same manner as GAHOS-1. However, GAHOS-2 goes
further by determining the best pump and storage tank to couple with the pumping schedule and SCR value.

The input file for GAHOS-2 (explained later in the chapter) contains a list of pumps and storage tanks that the program can consider (as options) in its analysis. The program simultaneously determines the optimal combination of pump, storage tank, SCR value and pumping schedule.

The chromosomes used in GAHOS-2 are very similar to those used in GAHOS-1. The only difference is that the later versions has more sectors in each chromosome and because of this is divided into two parts. The first part is 96 sectors long and is reserved for the pumping schedule. The second part is 3 sectors long and is reserved for the SCR value, the storage tank identification and pump identification.

Each and every pump and storage tank option considered in the optimization process is given an identification number. For example, if GAHOS-2 was given 6 separate pumps and 4 separate tanks to consider. Then pump identification numbers will range from 1 through 6, while the tank identification numbers will range from 1 through 4. For each and every chromosome in this example the following two properties must be true:

1. The 2\textsuperscript{nd} last sectors (i.e., 98\textsuperscript{th} sector) must contain a number between 1 and 6.
2. The last sector (i.e., 99\textsuperscript{th} sector) must contain a number between 1 and 4.
Fig. 4.1, shown on the next page presents a pictorial view of a GAHOS-2 chromosome. In Fig. 4.1, the SCR value is 0.547, and as can be seen the number presented in the 97th sector is acutely 547 (the product of 1000 x 0.547). This particular chromosome entails the use of storage tank number 2 and pump number 5.

Fig. 4.1: Chromosome structure of GAHOS-2

4.4 Evaluation

The evaluation of the chromosomes in GAHOS-1 is different than that of GAHOS-2. In GAHOS-1, the objective is to find the lowest possible daily electricity cost. This is accomplished by using equations 2.5 and 2.6 introduced in section 2.7 entitled “Discount rate, Inflation rate and Operating life”.

In GAHOS-2, the objective is to find the lowest possible project cost. For this purpose equations 2.3, 2.4, 2.5 and 2.6 are used. Evaluating the chromosomes of GAHOS-2 is more complicated than those of GAHOS-1. GAHOS-2 considers the initial purchase cost of the pump and storage tank. Also the O&M cost of the pump and storage
tank is considered. GAHOS-2 is based on the total project cost, thus the length of project (in years) can be of great influence on the characteristics of the optimal solution. As discussed in section 2.7, when evaluating a long term project it is best to use a present worth factor that assesses discount rate, inflation rate and off course the length of the project itself.

Both versions of the software evaluate the chromosomes over a 24 hour simulation. In these simulations smaller time steps have the ability to decrease the calculated costs. This occurs because smaller time steps allow the pumping schedules to be more flexible. However, the smaller the time step the higher the computational difficulty. The previous sentence is not meant as a deterrent for using small time steps (15 min. time steps as opposed to 60 min. time steps). The statement was included simply because it is true. As well, one of the main advantages of GAHOS over conventional optimization method (like dynamic programming or Lagrangean relaxation) is that these conventional method are crippled when required to use small time step (60 min. or less).

As previously discussed in this thesis, the fitness value of a chromosome reflects its ability to meet the objective(s) and constraints(s) associated with the system. Chromosomes that fail these constraints must be penalized by decreasing their fitness values. In both GAHOS-1 and GAHOS-2 the measured characteristic is cost, and it is the objective of these programs to find the lowest cost solution (i.e., optimal solution). Thus
the simplest way of penalizing chromosomes that fail to meet constraints is to increase their associated costs.

There are three different constraints defined in GAHOS:

1. The storage capacity of the water reservoir can not be exceeded.
2. The water demand must be meet at all times (i.e., the storage tank must always contain a sufficient quantity of water to satisfy the demand).
3. The simulation must terminate (i.e., end the 24 hr simulation) with at least an equal amount of water in the tank as it started with. In other words the SCR_0 must be greater or equal to the SCR_{24}.

The second and third constraints are more troublesome than the first. Constraints No. 2 and No. 3 are considered troublesome since by violating these constraints a chromosome can achieve a lower cost than by not violating them. Because lower cost chromosomes are superior to higher cost ones, the Genetic Algorithms naturally would favor (i.e., give an advantage to) the solution that violates constraints No. 2 and No. 3.

For instance, a chromosome violating constraint No. 3 can obtain very low electricity costs. A high SCR value can allow for the pumping schedule to reduce (if not eliminate) the quantity of times the pump is turned on. By not stipulating that the SCR_0 \geq SCR_{24}, the electricity cost of the first day (i.e., hours zero through 24) is reduced at the expense of the every day following.
A chromosome that is not penalized for violating constraint No. 2 may be at an advantage over a chromosome that has not violated the constraint. The supply of water to a population requires monetary expenditures. Therefore if a chromosome is not obliged (i.e., forced or penalized) to supply the necessary amount of water, than it would simply choose not to supply the water.

Constraint No.1 is different than its counterparts. It is physically impossible to exceed the volumetric capacity of a storage tank. Any chromosome that attempts (via its pumping schedule) to exceed the maximum capacity of the storage tank must be discouraged (i.e., stopped). In GAHOS-1 and GAHOS-2 this is done by simply eliminating that volume of water that surpasses the storage capacity (however the pumps were still running and consuming electricity at those times). By eliminating this excess volume of water the chromosome was not able to violate the laws of physics. There are non-violating chromosomes which can generate the same amount of water in storage (and in the same time frames) but at lower cost. Thus the violating chromosomes will naturally become extinct. Therefore there is no need to assign a monetary fine to this type of violation.

Constraints No. 2 and No. 3 must be monetarily penalized. These penalties are derived in such a way as to make it impossible for a violating chromosome to be economically competitive with other non-violating chromosome. For example, in the simulation described in Appendix 2, the daily electricity cost for even the worst non-violating chromosome was less than $200 (actually $151.03). The optimal chromosome
resulted in a daily pumping cost of only $53.68. Therefore, for this particular example, a violating chromosome must not be able to achieve (at a minimum) a pumping cost lower than $53.68. It would even be better if the violating chromosome was penalized to a point that its cost would be higher than even the most inefficient solution.

For the input files contained in the appendices of this thesis, the penalties levels were investigated and set to the following values: i) The software charges the chromosome a penalty of $2000/m³ for every cubic meter of demand not met. This analysis is done in every time step. ii) The software charges the chromosome a penalty of $200/m³ for every cubic meter of storage volume below the initial volume. This analysis is done once at the end of the 24 hour simulation.

The above three penalties are all located within the evaluation subroutine (called eval() ) of GAHOS-1 and GAHOS-2. The actual fines are defined within the computer code, thus the user in unable to directly altered these fines. However, the fines can be altered at the source code (i.e., the actual C++ code that is available in the appendices 8 and 9).

4.5 Reproduction

The basis of the reproduction procedure of GAHOS is the roulette wheel parent selection method. The roulette wheel parent selection method used in the software has two variables (as discussed previously in section Appendix 10). The first is the magnification factor (M) and the second is the magnification exponent (P).
The input parameters as defined in the appendices do not require a magnification factor (see section A10.3). However, a magnification factor might be needed if the input parameters significantly alter the costs associated with the chromosomes (i.e., alter the cost from their current values). Section A10.3 located in Appendix 10 describes a formula that should be used to determine if and/or what the M value should be. In both GAHOS-1 and GAHOS-2 the magnification factor is a constant defined within the code. In order to alter this constant the user must alter the definition of the M value within the evaluation subroutine (called eval(\(M\)).

The magnification exponent (\(M\)) is much more important than the magnification factor (\(P\)). This is because the \(P\) value controls the degree that the roulette wheel method is biased (see section A10.3). The value of \(P\) must be customized to the specific system in question. If the \(P\) value is too low, the software will take a long time to reach the optimal solution. Too high of a magnification exponent will cause floating point errors and fitness value truncations. The author suggests that \(P\) be defined between 10 and 15 while running GAHOS-1. While running GAHOS-2 the \(P\) value should be between 2 and 8 (Appendix 4).

The software developed in this thesis comes with a built-in safety procedure called \textit{absolute-best}. The procedure works like so: every generation, the chromosome with the lowest cost is replicated and its copy is stored for future use. Once a chromosome with a lower cost is found, the old chromosome is erased and its place is
taken by the new chromosome. Without the absolute-best function, the simulation outcome would be completely dependent on the results of the last generation. This is dangerous because many times the GA may find the optimal solution and then begin to oscillate. The absolute-best function ensures that the optimal solution (i.e., best solution found during the entire simulation) is returned regardless of the results of the final generation.

In order to encourage the software to investigate the specifics of the absolute best chromosome, the absolute best chromosome is introduced once into every generation prior to crossover. This is done by randomly erasing one chromosome and replacing it with a copy of the absolute best chromosome. If the previous generation naturally produced the absolute best chromosome then the absolute-best function is redundant because the roulette wheel method will reproduce that chromosome. However if the last generation did not have a high performing chromosome, then the absolute-best function will help the population recover.

**Fig. 4.2:** The effect of the absolute-best function.

![Graph showing the effect of absolute-best function](image-url)
As can be seen from Fig. 4.2, the absolute-best function can on average decrease the number of generation GAHOS requires to find the optimal solution (i.e., the $G_p$ value). It is important to remember that the term "average" was used in the above statement. This is because no two executions of GAHOS are exactly the same (i.e., they all utilize random number generators which are by definition random). Therefore it is false to state that a GA utilizing the absolute-best function will at all times outperform an identical GA without the absolute-best function. However, on average the absolute-best function will reduce the $G_p$ value and produce best solutions at any given generation (Appendix 16).

4.6 Mutation

Section 3.9 entitled “Bit Mutation” described two techniques: i) a conventional method called the CMM (Common Mutation Method). ii) A new and in many ways a superior technique called the ICMM (Individual Chromosome Mutation Method). The ICMM is used in both version of GAHOS and is sometimes complimented by the CMM. A discussion of this is available in the following paragraphs.

In both GAHOS-1 and GAHOS-2 the individual Chromosome Mutation Method is used to conduct mutation on the chromosome sectors representing the pump schedule (i.e., the first 96 sectors of the chromosome). The user inputs the mutation probability per chromosome ($P$), as well as the number of sectors (i.e., bits) to mutate per chromosome chosen for mutation ($N$) (reader may find it helpful to review section 3.8). A second mutation procedure called “bit switching” is utilized in parallel with the ICMM.
Bit switching was developed in this thesis specifically for GAs that incorporate scheduling. The idea stems from the fact that as the GA approaches the optimal solution the scheduling portion of the chromosome needs an extra boost that conventional mutation (even ICMM) methods can not deliver. As the chromosomes of GAHOS approach the optimal solution, their schedules may have the correct number of zeros and ones but a few of them are usually in the wrong location. "Bit switching" can sometimes alleviate this problem by randomly switching these bits.

An example of "bit switching" is given in the following paragraph. Imagine a pumping schedule that uses two hour time steps. The SCR value, tank size and pump discharge are dictated by the user. It is also determined that the pump must be running 20 hours per day (determined by comparing the average water demand rate and the pump capacity). The electric utility has set 14 peak hours (i.e., high electricity cost) and 10 off-peak hours (i.e., low electricity cost). Imagine that the GA is producing many near optimal chromosomes (i.e., two bits must exchange their values). These chromosomes are only using only 8 of the off-peak hours and thus the pump is on for 12 of the peak hours. These chromosomes would be optimal if they could somehow manage to reduce the peak pumping by two hours and at the same time increase the off-peak pumping by two hours.
Bit switching allows for pairs of bits to switch identities as shown in Fig. 4.3. This works well for near optimal schedules since it does not change the numbers of zeros and ones, only a few of their locations (i.e., one or two pairs of bits are switched).

Fig. 4.3: The Bit Switching procedure

Bit switch this pair

\[ \begin{array}{cccccccc}
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array} \]

Resulting chromosome

\[ \begin{array}{cccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array} \]

Bit switching aids in scheduling where there is an off-peak and peak periods. Also in cases where there is a cost associated with turning on a pump, bit switching will consolidate the pumping bits of the chromosome (i.e., attempt to group all the 1 bits together in the most economical way).

In both GAHOS-1 and GAHOS-2 the user must specify the number of pairs to switch. Of course not all the chromosomes in a population must undergo bit switching, only a few chromosomes are chosen for this procedure. Thus there is a specified probability per chromosome.

So far the reader has been presented with a discussion of how ICMM and bit switching are used on the scheduling bits of a chromosome. However the chromosome
retains information on more than just the pumping schedule. For example, the chromosomes used in GAHOS-1 hold data on the SCR value and those of GAHOS-2 hold data on the SCR, pump type and storage type. These bits must also undergo mutation. For these sectors the common bit mutation method is used. In order to simplify the code, a single bit mutation probability represents all 3 data types.

The next section will focus on the manner in which input is presented to the computer programs. The input and output files of both GAHOS-1 and GAHOS-2 will be discussed and references made to examples of these files (available in the appendices).

4.7 Input and Output

The input data required to run both versions of GAHOS are divided into two forms: i) input file and ii) data entered via the keyboard. The input file specifications for GAHOS-1 differ from that of GAHOS-2. This is also true for the data entered via the keyboard.

The input file of GAHOS-1 (see Appendix 14) begins with 24 numbers (one number per line) representing the electricity demand data in units of $/kW-hr. The first electricity cost figure represents the cost of consuming electricity at time zero (first hour of simulation), while the 24th number (i.e., line) represents the electricity cost for the last hour of the simulation. The next 24 lines (one number per line) represent the water demand rates for time zero through 24 (the units used are m$^3$/s). This is the complete set
of data (in input file form) needed to run GAHOS-1, note that the file is only 48 lines long.

The input file for GAHOS-2 (see Appendix 15), uses the same format as that of GAHOS-1. However, additional lines of data are required (lines 49, 50 and higher). Line 49 is actually a blank line used to divide the above data from the storage tank data discussed below. Every storage tank that GAHOS-2 must consider is described by 3 parameters located in a single line starting at line 50. The first parameter is the capital cost (i.e., in current value Canadian dollars) followed by the O&M cost for the first year (represented as a percentage of the capital cost). The third parameter is storage capacity in m$^3$. The reason behind the use of these parameters are available in section 2.7.

After the storage tank data is completed comes another blank line followed immediately by the pump parameter data. Just like the tanks, every pump is described in just one line (however using 5 parameters not 3). The parameters in order from left to right are: capital cost, O&M cost for the first year represented as a percentage of the capital cost, pump flow in m$^3$/s, head in m, and pump efficiency represented as a fraction not a percentage.

The data prompt (requested or asked) by the software is very simple to understand. This includes the input file name, time step size, population size, number of generations required and mutation parameters. It is important to note that the input file must be in a simple text only format (i.e., unformatted). The reader can make sure of
this by editing the file in a word processing software (i.e., MS Word, Word Perfect or even MS DOS file editor) and saving it as a text file (i.e., "txt"). However once this file is saved as a "txt" file, the name of the file can change. For example "data.txt" can be changed into "Toronto.may". GAHOS does not need the input file name to end with "txt".

4.8 Model Validation

The customary method of validating an optimization model is to produce what is called an exhaustive enumeration search. This search requires the creation and evaluation of every possible solution. GAHOS-1 was validated using exhaustive enumeration, however due to practical reasons GAHOS-2 could not be validated this way (as will be discussed below).

GAHOS-1 and GAHOS-2 are much like a flight of stairs. Each version is based on the structure of the previous one, yet each new program incorporates more complex optimization objectives. For example, the original version of GAHOS-1 (called GAHOS-1A) optimized the pumping schedule for a given set of conditions; a specified pump, a specified storage tank and a specified SCR value. The final version of GAHOS-1 (called GAHOS-1B) did not need a specified SCR value since it could optimize it (i.e., SCR) along with the pumping schedule. GAHOS-2 could optimize the pumping schedule, SCR value, and choose the best combination of pump and storage tank.
The model validation process was conducted in three separate stages. The first stage is based on GAHOS-1A. The second stage is based on GAHOS-1B, while the third stage is based on GAHOS-2. As previously mentioned, each version of the software is based on its predecessor. Therefore components that have been validated in previous versions were not revalidated.

Table 4.0: A summary of the three versions of GAHOS used in the validation

<table>
<thead>
<tr>
<th>Name of Version</th>
<th>Main Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAHOS-1A</td>
<td>Determines the optimal pumping schedule. User must define the SCR value.</td>
</tr>
<tr>
<td>GAHOS-1B</td>
<td>Determines the optimal pumping schedule. The software also determines the best SCR value.</td>
</tr>
<tr>
<td>GAHOS-2</td>
<td>Determines the optimal pumping schedule, SCR value, as well as the best pump and tank types. The user must define (in an input file) an inventory of possible pump types and pump types.</td>
</tr>
</tbody>
</table>

4.8.1 First Stage

GAHOS-1A could only optimize the pumping schedule. It required the user to input the SCR as well as the pump and storage tank parameters. The pumping schedule could be partitioned between 15 minute to one hour segments. In other words the scheduling chromosome could be between 24 to 96 sectors long depending on the time step specified. The program was executed using a one hour time step based on the average day demand rates (called AVRDAY-1 and discussed in section 2.2 and in Appendix 6).

A special computer code called ENUMERO (named as such because it is based on complete enumeration) was constructed to test every possible combination of pumping
schedule and return the chromosome with the lowest pumping cost (Appendix 12).

ENUMERO created all $2^{24}$ combination of possible chromosomes using binary addition, an example of this is demonstrated in Fig. 4.4 shown below.

**Fig. 4.4:** An example of binary addition

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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 1 1 1 1 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 1 1 1 1 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 1 1 1 0
```

These procedures took several hours and could not have been possible if the time step was much smaller than 15 or 20 minutes. The structure of ENUMERO is very simple since it only has two main duties: i) The creation of chromosomes using binary addition and ii) The evaluation of every chromosome created. ENUMRO is considered to be a "dumb algorithm" since the vast majority of the chromosomes it produces violate the hydraulic constraints.

### 4.8.2 Second Stage

The scheduling capability of the software has been verified in the first stage (GAHOS-1A). The next step is to test the capability of GAHOS to optimize the pump scheduling and SCR value simultaneously. This version of the software is called GAHOS-1B.
The verification of GAHOS-1B is done by running GAHOS-1A approximately 200 times using the entire range of SCR values, from 0 to 1.00 (see Appendix 1). This experiment identified the optimal range of SCR values ratios and the corresponding daily pumping cost. The result are shown in Appendix 1. By examining the solution produced by GAHOS-1B, it was evident that the program returned the SCR value which was in the optimal range as well as with corresponding pumping schedules that generated the minimum pumping costs.

4.8.3 Third Stage

Based on the results in the first two stages, one can be confident that GAHOS-1B can optimize both the pumping schedule and the SCR value. GAHOS-2 is even more advanced than GAHOS-1B (and of-course GAHOS-1A) since it can optimize the pump and storage tank as well. The question arises as to how to validate this version?

GAHOS-2 was validated by running GAHOS-1B a total of 25 times. It was decided to examine the results from 5 different pumps and 5 different storage tanks (i.e. 25 combinations). The storage tank and pump options are shown below in Tables 4.1 and 4.2 respectively.

**Table 4.1:** Cost properties for the five different storage tanks examined

<table>
<thead>
<tr>
<th>Designation</th>
<th>Storage size [m$^3$]</th>
<th>Principal Cost [£]</th>
<th>M &amp; O Cost [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank 1</td>
<td>2000</td>
<td>580000</td>
<td>7.0</td>
</tr>
<tr>
<td>Tank 2</td>
<td>4000</td>
<td>860000</td>
<td>6.9</td>
</tr>
<tr>
<td>Tank 3</td>
<td>7000</td>
<td>1280000</td>
<td>6.8</td>
</tr>
<tr>
<td>Tank 4</td>
<td>12000</td>
<td>1980000</td>
<td>6.7</td>
</tr>
<tr>
<td>Tank 5</td>
<td>18000</td>
<td>2820000</td>
<td>6.6</td>
</tr>
</tbody>
</table>
The lowest cost combination is tank #1 with pump #4. GAHOS-2 is able to come to this same conclusion in less than 2 minutes (on an IBM 6x86, 133 MHz processor). GAHOS-1B provided the daily electricity cost for each of the 25 combinations. Then by hand the capital cost and O&M costs were added.

Table 4.2: Cost properties for the five different pumps examined

<table>
<thead>
<tr>
<th>Designation</th>
<th>Principal Cost [$]</th>
<th>M &amp; O Cost [%]</th>
<th>Flow [m$^{3}$/s]</th>
<th>Head [m]</th>
<th>Efficiency [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pump 1</td>
<td>1,116,090</td>
<td>10</td>
<td>0.191</td>
<td>60.3</td>
<td>78</td>
</tr>
<tr>
<td>Pump 2</td>
<td>1,023,706</td>
<td>13</td>
<td>0.180</td>
<td>53.9</td>
<td>81</td>
</tr>
<tr>
<td>Pump 3</td>
<td>1,178,509</td>
<td>9</td>
<td>0.203</td>
<td>62.1</td>
<td>77</td>
</tr>
<tr>
<td>Pump 4</td>
<td>1,231,594</td>
<td>8</td>
<td>0.231</td>
<td>55.3</td>
<td>76</td>
</tr>
<tr>
<td>Pump 5</td>
<td>1,193,829</td>
<td>11</td>
<td>0.196</td>
<td>68.2</td>
<td>79</td>
</tr>
</tbody>
</table>

The project life was assumed to be 20 years with an inflationary constant of 3% and a discount rate of 10% (realistic for engineering projects, White et al. (1989)). The results are shown in Table 4.3, below.

The type of validation conducted on GAHOS-2 has a few uncertainties associated with it. GAHOS-1A and GAHOS-1B reached the optimal solution (with 100% certainty) during the validation process. However, this might not always be the case. All Genetic Algorithms suggest that the most optimal chromosome is the one “discovered” during the simulation. Yet there is no guaranty that the chromosome suggested by the GA is actually the mathematical optimal solution of the system. Even if the number of chromosomes and quantity of generations were set equal to absurdly large values.
Table 4.3:  Total System Cost for all possible combinations

<table>
<thead>
<tr>
<th>Combination</th>
<th>Electricity Cost [$/day]</th>
<th>Total System Cost [$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank 1 - Pump 1</td>
<td>77.53</td>
<td>3,582,489</td>
</tr>
<tr>
<td>Tank 1 - Pump 2</td>
<td>66.85</td>
<td>3,673,742</td>
</tr>
<tr>
<td>Tank 1 - Pump 3</td>
<td>81.44</td>
<td>3,601,887</td>
</tr>
<tr>
<td>Tank 1 - Pump 4</td>
<td>73.42</td>
<td>3,545,599</td>
</tr>
<tr>
<td>Tank 1 - Pump 5</td>
<td>85.99</td>
<td>3,898,475</td>
</tr>
<tr>
<td>Tank 2 - Pump 1</td>
<td>58.79</td>
<td>3,986,837</td>
</tr>
<tr>
<td>Tank 2 - Pump 2</td>
<td>51.45</td>
<td>4,090,859</td>
</tr>
<tr>
<td>Tank 2 - Pump 3</td>
<td>60.66</td>
<td>3,998,460</td>
</tr>
<tr>
<td>Tank 2 - Pump 4</td>
<td>54.02</td>
<td>3,947,456</td>
</tr>
<tr>
<td>Tank 2 - Pump 5</td>
<td>64.51</td>
<td>4,292,391</td>
</tr>
<tr>
<td>Tank 3 - Pump 1</td>
<td>53.68</td>
<td>4,676,817</td>
</tr>
<tr>
<td>Tank 3 - Pump 2</td>
<td>51.64</td>
<td>4,801,068</td>
</tr>
<tr>
<td>Tank 3 - Pump 3</td>
<td>53.10</td>
<td>4,679,113</td>
</tr>
<tr>
<td>Tank 3 - Pump 4</td>
<td>42.39</td>
<td>4,612,544</td>
</tr>
<tr>
<td>Tank 3 - Pump 5</td>
<td>58.65</td>
<td>4,979,526</td>
</tr>
<tr>
<td>Tank 4 - Pump 1</td>
<td>53.68</td>
<td>5,649,377</td>
</tr>
<tr>
<td>Tank 4 - Pump 2</td>
<td>51.64</td>
<td>5,977,819</td>
</tr>
<tr>
<td>Tank 4 - Pump 3</td>
<td>51.21</td>
<td>5,848,657</td>
</tr>
<tr>
<td>Tank 4 - Pump 4</td>
<td>42.39</td>
<td>5,789,294</td>
</tr>
<tr>
<td>Tank 4 - Pump 5</td>
<td>58.65</td>
<td>6,156,276</td>
</tr>
<tr>
<td>Tank 5 - Pump 1</td>
<td>53.68</td>
<td>7,252,250</td>
</tr>
<tr>
<td>Tank 5 - Pump 2</td>
<td>51.64</td>
<td>7,376,501</td>
</tr>
<tr>
<td>Tank 5 - Pump 3</td>
<td>51.21</td>
<td>7,247,339</td>
</tr>
<tr>
<td>Tank 5 - Pump 4</td>
<td>42.39</td>
<td>7,187,976</td>
</tr>
<tr>
<td>Tank 5 - Pump 5</td>
<td>58.65</td>
<td>7,554,959</td>
</tr>
</tbody>
</table>

When validating GAHOS-2, the author created a problem with a total of only 25 combinations of pumps and storage tanks. Each of these combinations was tested using GAHOS-1B. The 25 different optimal solutions where compared and the lowest cost solution was taken to be the “optimal one”. GAHOS-2 was run once and considered all 5 pumps and 5 tanks. GAHOS-2 suggested the same combination of pump, tank, pumping schedule and SCR value that the 25 GAHOS-1B executions did. However, imagine that by some chance one of the 25 separate executions was defected, and the true best solution...
was not found. The fact that this chance exists is the major flaw of the validation used on GAHOS-2.

Although some may be uncomfortable with this type of validation, an exhaustive enumeration would have been impossible. For example an exhaustive enumeration on a problem with 5 pump options, 5 tank options, 15 min. time steps and an S/C ratio partitioned in 5% intervals would result in $3.961 \times 10^{31}$ possible combinations. Even if a computer could evaluate one billion of these combinations in a second, it would take $1.256 \times 10^{15}$ years to validate the code.

4.9 Summary

Chapter 4 presented to the user information pertaining to the: chromosome structure, evaluation subroutine (including penalties), reproduction subroutines and mutation subroutines. The chapter also introduced the reader to the three stages used in the model validation process. The key conclusions that can be drawn from chapter 4 are list below:

- The chromosomes of GAHOS-1 are 97 sectors long
- The chromosomes of GAHOS-2 are 99 sectors long
- The time steps used may vary between 15 and 60 minutes
- Penalizing chromosomes that fail to meet the constraints of the system is crucial
- The absolute-best function created in this thesis can increase the probability that GAHOS can find the optimal solution
- GAHOS uses three mutation methods in parallel: ICMM, CMM and “bit switching”
The following chapter (i.e., chapter 5) focuses on two proposals for the further advancement of GAHOS. The information presented in chapter 4 will aid the reader in the comprehension of the recommendations and proposals presented in chapter 5.
5. FUTURE VERSIONS OF GAHOS

5.1 Introduction

This last chapter is meant to provide the reader with suggestions for further research using the GAHOS platform. Two proposals are presented for developing more advanced versions of the computer software (GAHOS-3 and GAHOS-4).

5.2 GAHOS-3 Blueprints

GAHOS-1 and GAHOS-2 incorporated a pumping system consisting of a single pump. However in reality most pumping systems consist of a battery of pumps. These are typically located in a parallel format (Walski and Ormsbee, 1989). The proposed pumping system which could be incorporated into GAHOS-3 is shown below as Fig. 5.0.

Fig. 5.0: GAHOS-3 pumping system

As can be seen from the illustration, there are only two types of pumps. Before each pump there is a valve that controls whether or not the water will pass through the pump. There is also a valve located at the junction where the flow is divided into the parallel configuration. This particular valve has three distinct settings; it may open both branches of the parallel pipes, only one or none. If both branches are open than the
operation of the pumps must be identical for both. This is because the total dynamic head (TDH) produced by the pumps (in each branch) must be identical in order to unite the flow again (Hirrel, 1989).

There are a total of 7 possible combinations of pumps. Three for when both branches are open, three for when only one branch is open (it does not matter which of the branches is closed and which one is open) and one for when the entire system is closed. These seven combinations were based on the assumption that each of the two pump types have only a single operating point. In other words a specific head, discharge and energy efficiency ratio. Let us assume that there are actually 2 operating points for each of the two pumps. In other words a high and a low settings.

<table>
<thead>
<tr>
<th>Pump # 1</th>
<th>Pump # 2</th>
<th>Branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>High</td>
<td>Both</td>
</tr>
<tr>
<td>Off</td>
<td>Low</td>
<td>Both</td>
</tr>
<tr>
<td>Low</td>
<td>Off</td>
<td>Both</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>Both</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>Both</td>
</tr>
<tr>
<td>High</td>
<td>Off</td>
<td>Both</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>Both</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>Both</td>
</tr>
<tr>
<td>Off</td>
<td>High</td>
<td>One</td>
</tr>
<tr>
<td>Off</td>
<td>Low</td>
<td>One</td>
</tr>
<tr>
<td>Low</td>
<td>Off</td>
<td>One</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>One</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>One</td>
</tr>
<tr>
<td>High</td>
<td>Off</td>
<td>One</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>One</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>One</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>Irrelevant</td>
</tr>
</tbody>
</table>

This would result in 17 possible pump combinations. As can be seen there are eight possible pumping combinations for the single branch and another eight possible pumping
combinations for the parallel configuration. The last (i.e., 17th) pumping combination is for when all pumps are turned off.

Unlike previous versions of the computer program, GAHOS-3 would incorporate both a time-of-use rate structure and the previously unused peak-demand rate structure. The program could evaluate both rate structures, but would only utilize the one with the higher cost. GAHOS-1 and GAHOS-2 did not use the peak-demand rate structure since this type of rate structure does not make sense for a pumping system employing a single pump.

The chromosome structure will be identical to that of GAHOS-1. The only difference is that instead of simple "zeros" and "ones", the range of number would be between zero and 16. A separate array (of 17 sections) will store the electricity consumption data for the 17 possible combinations.

It is the author's estimate that the recoding of GAHOS-1 into GAHOS-3 would take between 15-35 hours. The verification process on the other hand will take significant computer resources and might be impossible to do. This is because of the large number of pump combinations which have a profound effect on the number of possible pumping schedules. For example, for the system shown in Fig 5.0 using single operating point pumps (i.e., 7 possible pumping combination), one hour time steps and using 5% separation for the S/C ratio, this would result in $3.83 \times 10^{21}$ (or $7^{24} \times 20$) possible pumping schedules.
The reader should note that GAHOS-3 would not be limited to only the pumping system illustrated in Fig. 5.0. It is up to the user to decide on the pumping system configuration. The software only views the pumping system as an array of possible pumping combinations. Each of these combinations is represented by three numbers; the total discharge, head and electricity consumption.

5.3 GAHOS-4 Blueprints

Even though GAHOS-3 can manage complicated pumping system that use multiple operating points, it would not be able to comprehend the complex nature of the distribution network it supplies water to. Presuming that an elevated storage tank will meet the pressure requirements for all nodes in the network is a poor and unrealistic assumption (Chao, 1979). The presence of this assumption in GAHOS-3 (as well as in GAHOS-1 and GAHOS-2) undermines the utility of software.

A new software version called GAHOS-4 would attempt to address the inadequacies of its predecessors. GAHOS-4 would be identical to GAHOS-3 as far as chromosome structure, mutation, crossover and pumping configuration options. The only difference lays in the evaluation subroutine. GAHOS-4 would utilize a steady state network solver for the purpose of gathering data on all nodal pressures.

The Network Solver (NS) does not need a sophisticated user interface because it is not in direct contact with the user. However, the NS has to be very fast since it is used many times for each chromosome. GAHOS-4 creates an input file for the NS that
includes information such as the pumping system configuration (i.e., which pumps are on and at what speed), nodal demands and pipe properties. Once the NS has assessed the system, it produces an output file that contains information such as pipe velocities, pipe head loss and nodal pressure. GAHOS-4 then analyzes the output file and assigns penalties for constraints not meet (i.e., inadequate nodal pressure).

The size of time step used in the pumping schedule dictates the number of times the NS must be used before the chromosome is assigned a final fitness value. For example, a chromosome using a 30 min. time step in its pumping schedule and one hour time steps for the nodal demand data will require 48 separate evaluations by the NS. This occurs because every time the pumping system or demand profile is changed, GAHOS-4 must ensure that the constraints are met. It is the smallest time step of either the pumping schedule or nodal demand profile that dictates the size of time step used in the evaluation subroutine (i.e., the number of times the NS is used per chromosome).

It is generally the case that the pumping schedule time step is smaller than the time step used in the nodal demand profile. This is typically do to the fact that gathering nodal demand data is expensive and thus is not readily available. The pumping schedule time step on the other hand is simply a question of computer resources. Maximum benefit can be drawn from the software when the time steps are equal.

There are two ways in which to incorporate the NS in GAHOS-4. The first is to insert the entire NS code into the GAHOS-4 code. The second method is to define the NS as a separate executable entity, which is called (i.e., activated) by GAHOS-4. The
first method had the advantage of having the entire software within one file which results in quicker communication between the different subroutines. However, the disadvantage is that the NS code must be altered in order to be compatible with the rest of the GAHOS-4 program. Method number two requires no changes to be made to the NS. However, the communication between the NS and GAHOS-4 is done using input and output files which greatly increase the execution time of the total program.

Even though a steady state network solver was suggested for use in GAHOS-4, other types of NS can also be used. A more accurate type of NS would be one that considers transient flow as opposed to simply steady state flow (Frey et al., 1996). The type of NS used in GAHOS-4 should reflect the sort of analysis the user is interested in.
6. CONCLUSIONS AND RECOMMENDATIONS

The focus of this thesis was the creation of a computer software program that utilizes the theories of Genetic Algorithms for the purpose of optimizing a water distribution systems. The computer program was called GAHOS (Genetic Algorithm Hydraulic Optimization Software) and applications of two versions of this program were presented.

Below is a list of the conclusions that resulted from the research conducted in this thesis. These conclusions represent both general statements dealing with Genetic Algorithms and conclusions based on GAHOS.

1. The speed with which a Genetic Algorithms can reach the optimal solution is highly dependent on the time required to evaluate an individual solution. A slow evaluation subroutine (i.e., computationally extensive) can render a Genetic Algorithms impractical for real time analysis.

2. Genetic Algorithms including GAHOS-1 that are designed to execute in real time must utilize a fast evaluation procedure (a minimum of 2 chromosome evaluations per second).

3. GA’s including GAHOS must be able to proficiently use “penalties” in order to reach the optimal chromosome quickly. Penalties are the only means that a GA has to restrict the use of chromosomes that violate the systems constraint(s).

4. The fitness value is a method by which a GA can promote high performing chromosomes.
5. All GAs must have their mutation, crossover, and parent selection procedures custom made for a specific system.

6. The parameters required for the above three procedures must be tested (i.e., configured) to the system using input data that is similar to that expected in practice.

7. The population size (i.e., number of chromosomes in each generation) and the number of generations required are two important parameters. They have a profound effect on the probability of finding an optimal solution in a simulation.

8. The parameter used in GAs are optimized to a system and its input data for the purpose of decreasing the $G_P$ value. These parameters also have a profound effect on the confidence associated with the results produced by a GA at the end of its execution.

Recommendations for future work using GAHOS are summarized below:

1. GAHOS-1 should be redesigned to incorporate both a time-of-use electricity rate structure and the previously unused peak-demand rate structure. The pumping system should also be enhanced by using multiple pumps (using pumps in series and parallel). This recommended version was entitled GAHOS-3.

2. GAHOS-3 can be further enhanced by using a network solver to evaluate how the chromosomes meet the hydraulic constraints of the water distribution network. The installation of the network solver was entitled “GAHOS-4 Blueprints”.
REFERENCES


Wong, K.P., and Y.W. Wong,  *Genetic And Genetic/Simulated-Annealing Approaches to*
Appendix 1 - Effect of SCR on Daily Electricity Cost

In order to understand the effect the SCR has on the operational cost of the hydraulic system, an experiment was undertaken to provide data that would correlate the operational cost vs. SCR. Since varying the SCR does not have an effect on any cost parameters (i.e. maintenance, depreciation, discount rate etc.) except the electricity cost. It was therefore numerically efficient to simply examine the effect the SCR has on the daily electricity costs.

The table below is a summaries one such experiment. The experiment used the above shown hydraulic system and GA parameters that were used as input data for GAHOS-1A. The electricity cost data and water demand rates were provided by the input file 15-bit.txt shown in Appendix 14.

Hydraulic System Parameters:

Pump: flow = 0.191 [m$^3$/s], head = 60.3 [m], efficiency = 78%

Storage tank: storage capacity = 12,000 [m$^3$]

GA parameters:

Number of generations: 300

Population: 100

Time step: 15 min.

Probability of mutation: 0.09 or 9 % / Population / Generation

Switches per chromosome: 2
Table A1-1: Starting storage volume with corresponding Daily electricity cost for a 12,000 m³ tank.

<table>
<thead>
<tr>
<th>S/C ratio [%]</th>
<th>Starting Storage [m³]</th>
<th>Daily Electricity Cost [$/day]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>60.5</td>
</tr>
<tr>
<td>0.01</td>
<td>1.2</td>
<td>60.5</td>
</tr>
<tr>
<td>0.01</td>
<td>1.2</td>
<td>58.8</td>
</tr>
<tr>
<td>0.1</td>
<td>12</td>
<td>58.8</td>
</tr>
<tr>
<td>0.25</td>
<td>30</td>
<td>58.8</td>
</tr>
<tr>
<td>0.5</td>
<td>60</td>
<td>58.8</td>
</tr>
<tr>
<td>0.5</td>
<td>60</td>
<td>53.7</td>
</tr>
<tr>
<td>5</td>
<td>600</td>
<td>53.7</td>
</tr>
<tr>
<td>67</td>
<td>8040</td>
<td>53.7</td>
</tr>
<tr>
<td>68</td>
<td>8160</td>
<td>53.7</td>
</tr>
<tr>
<td>67</td>
<td>8040</td>
<td>53.7</td>
</tr>
<tr>
<td>68</td>
<td>8160</td>
<td>53.7</td>
</tr>
<tr>
<td>69</td>
<td>8280</td>
<td>55.4</td>
</tr>
<tr>
<td>70</td>
<td>8400</td>
<td>55.4</td>
</tr>
<tr>
<td>70.5</td>
<td>8460</td>
<td>57.1</td>
</tr>
<tr>
<td>71</td>
<td>8520</td>
<td>57.1</td>
</tr>
<tr>
<td>72</td>
<td>8640</td>
<td>58.8</td>
</tr>
<tr>
<td>73</td>
<td>8760</td>
<td>58.8</td>
</tr>
<tr>
<td>74</td>
<td>8880</td>
<td>60.5</td>
</tr>
<tr>
<td>74.5</td>
<td>8940</td>
<td>60.5</td>
</tr>
<tr>
<td>75</td>
<td>9000</td>
<td>62.2</td>
</tr>
<tr>
<td>76</td>
<td>9120</td>
<td>62.2</td>
</tr>
<tr>
<td>76.5</td>
<td>9180</td>
<td>63.9</td>
</tr>
<tr>
<td>77</td>
<td>9240</td>
<td>63.9</td>
</tr>
<tr>
<td>77.5</td>
<td>9300</td>
<td>63.9</td>
</tr>
<tr>
<td>78</td>
<td>9360</td>
<td>65.6</td>
</tr>
<tr>
<td>78.5</td>
<td>9420</td>
<td>65.6</td>
</tr>
<tr>
<td>79</td>
<td>9480</td>
<td>67.3</td>
</tr>
<tr>
<td>80</td>
<td>9600</td>
<td>67.3</td>
</tr>
<tr>
<td>80.5</td>
<td>9660</td>
<td>69</td>
</tr>
<tr>
<td>81</td>
<td>9720</td>
<td>69</td>
</tr>
<tr>
<td>82</td>
<td>9840</td>
<td>70.7</td>
</tr>
<tr>
<td>83</td>
<td>9960</td>
<td>70.7</td>
</tr>
<tr>
<td>84</td>
<td>10080</td>
<td>72.4</td>
</tr>
<tr>
<td>84.5</td>
<td>10140</td>
<td>72.4</td>
</tr>
<tr>
<td>85</td>
<td>10200</td>
<td>74.1</td>
</tr>
<tr>
<td>86</td>
<td>10320</td>
<td>74.1</td>
</tr>
<tr>
<td>86.5</td>
<td>10380</td>
<td>75.8</td>
</tr>
<tr>
<td>87</td>
<td>10440</td>
<td>75.8</td>
</tr>
<tr>
<td>88</td>
<td>10560</td>
<td>77.5</td>
</tr>
<tr>
<td>88.5</td>
<td>10620</td>
<td>77.5</td>
</tr>
<tr>
<td>89</td>
<td>10680</td>
<td>79.2</td>
</tr>
<tr>
<td>90</td>
<td>10800</td>
<td>79.2</td>
</tr>
<tr>
<td>91</td>
<td>10920</td>
<td>80.9</td>
</tr>
<tr>
<td>91.5</td>
<td>10980</td>
<td>80.9</td>
</tr>
<tr>
<td>92</td>
<td>11040</td>
<td>82.6</td>
</tr>
<tr>
<td>93</td>
<td>11160</td>
<td>82.6</td>
</tr>
</tbody>
</table>
Analyzing the data available in Table A1-1 one can conclude that an optimal SCR range exists (between 5% and 68%). By repeating the experiment on several storage tank sizes the construction of Table A1-2 was possible. The optimal SCR ranges for the 100, 2000, 4000 and 8000 m³ storage tanks are shown in Table A1-3, Table A1-4, Table A1-5 and Table A1-6 respectively. The experiments on these storage tanks used the same input data that was documented at the start of this appendix.

Table A1-2: Optimal SCR range for several storage tank sizes

<table>
<thead>
<tr>
<th>Maximum Storage Capacity [m³]</th>
<th>Optimal SCR range [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>90 - 95</td>
</tr>
<tr>
<td>2000</td>
<td>0.5 - 30</td>
</tr>
<tr>
<td>4000</td>
<td>5 - 10</td>
</tr>
<tr>
<td>8000</td>
<td>10 - 50</td>
</tr>
<tr>
<td>12000</td>
<td>5 - 68</td>
</tr>
</tbody>
</table>
Table A1-3: Optimal SCR range for a 100 m³ Capacity Tank

<table>
<thead>
<tr>
<th>SCR [%]</th>
<th>Daily Pumping Cost [$/day]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>133.4</td>
</tr>
<tr>
<td>10</td>
<td>133.4</td>
</tr>
<tr>
<td>20</td>
<td>133.4</td>
</tr>
<tr>
<td>30</td>
<td>133.4</td>
</tr>
<tr>
<td>40</td>
<td>133.4</td>
</tr>
<tr>
<td>50</td>
<td>133.4</td>
</tr>
<tr>
<td>60</td>
<td>133.4</td>
</tr>
<tr>
<td>70</td>
<td>133.4</td>
</tr>
<tr>
<td>80</td>
<td>133.4</td>
</tr>
<tr>
<td>85</td>
<td>133.4</td>
</tr>
<tr>
<td>90</td>
<td>132.6</td>
</tr>
<tr>
<td>95</td>
<td>132.6</td>
</tr>
<tr>
<td>100</td>
<td>133.4</td>
</tr>
</tbody>
</table>

Table A1-4: Optimal SCR range for a 2000 m³ Capacity Tank

<table>
<thead>
<tr>
<th>SCR [%]</th>
<th>Daily Pumping Cost [$/day]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>78.3</td>
</tr>
<tr>
<td>0.05</td>
<td>78.3</td>
</tr>
<tr>
<td>0.5</td>
<td>77.5</td>
</tr>
<tr>
<td>2.5</td>
<td>77.5</td>
</tr>
<tr>
<td>5</td>
<td>77.5</td>
</tr>
<tr>
<td>10</td>
<td>77.5</td>
</tr>
<tr>
<td>20</td>
<td>77.5</td>
</tr>
<tr>
<td>30</td>
<td>77.5</td>
</tr>
<tr>
<td>40</td>
<td>79.2</td>
</tr>
<tr>
<td>50</td>
<td>80.9</td>
</tr>
<tr>
<td>60</td>
<td>84.3</td>
</tr>
<tr>
<td>70</td>
<td>86.1</td>
</tr>
<tr>
<td>80</td>
<td>87.8</td>
</tr>
<tr>
<td>90</td>
<td>89.5</td>
</tr>
<tr>
<td>100</td>
<td>91.2</td>
</tr>
</tbody>
</table>
Table A1-5: Optimal SCR range for a 4000 m³ Capacity Tank

<table>
<thead>
<tr>
<th>SCR [%]</th>
<th>Daily Pumping Cost [$/day]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>60.5</td>
</tr>
<tr>
<td>0.125</td>
<td>58.8</td>
</tr>
<tr>
<td>0.25</td>
<td>58.8</td>
</tr>
<tr>
<td>1.25</td>
<td>58.8</td>
</tr>
<tr>
<td>2.5</td>
<td>58.8</td>
</tr>
<tr>
<td>5</td>
<td>57.1</td>
</tr>
<tr>
<td>6.25</td>
<td>57.1</td>
</tr>
<tr>
<td>7.5</td>
<td>57.1</td>
</tr>
<tr>
<td>10</td>
<td>57.1</td>
</tr>
<tr>
<td>20</td>
<td>60.5</td>
</tr>
<tr>
<td>30</td>
<td>63.9</td>
</tr>
<tr>
<td>40</td>
<td>67.3</td>
</tr>
<tr>
<td>50</td>
<td>72.4</td>
</tr>
<tr>
<td>60</td>
<td>75.8</td>
</tr>
<tr>
<td>70</td>
<td>79.2</td>
</tr>
<tr>
<td>80</td>
<td>84.3</td>
</tr>
<tr>
<td>90</td>
<td>87.8</td>
</tr>
<tr>
<td>100</td>
<td>91.2</td>
</tr>
</tbody>
</table>

Table A1-6: Optimal SCR range for a 8000 m³ Capacity Tank

<table>
<thead>
<tr>
<th>SCR [%]</th>
<th>Daily Pumping Cost [$/day]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>60.5</td>
</tr>
<tr>
<td>0.125</td>
<td>58.8</td>
</tr>
<tr>
<td>0.625</td>
<td>58.8</td>
</tr>
<tr>
<td>1.25</td>
<td>58.8</td>
</tr>
<tr>
<td>10</td>
<td>53.7</td>
</tr>
<tr>
<td>20</td>
<td>53.7</td>
</tr>
<tr>
<td>30</td>
<td>53.7</td>
</tr>
<tr>
<td>40</td>
<td>53.7</td>
</tr>
<tr>
<td>50</td>
<td>53.7</td>
</tr>
<tr>
<td>55</td>
<td>55.4</td>
</tr>
<tr>
<td>60</td>
<td>60.5</td>
</tr>
<tr>
<td>70</td>
<td>67.3</td>
</tr>
<tr>
<td>80</td>
<td>75.8</td>
</tr>
<tr>
<td>90</td>
<td>84.3</td>
</tr>
<tr>
<td>100</td>
<td>91.2</td>
</tr>
</tbody>
</table>
Appendix 2 - Effect of magnification exponent on Daily Electricity Cost

The following data was gathered by running GAHOS-1A under different magnification exponents. This version of GAHOS is only allowed to optimize the schedule and S/C ratio. The input file 15-bit.txt (shown in Appendix 14) was used in combination with the hydraulic system and GA parameters defined below.

The analysis done in this appendix is required when calibrating a GA. Magnification exponents and other GA parameters must be calibrated in order to optimize the performance of the GA. Calibrating these parameters is a simple exercise of test the performance of the GA using different parameter values (i.e. magnification exponent of 20, 15, 10 or 5?).

Hydraulic System Parameters:
Pump: flow = 0.191 [m$^3$/s], head = 60.3 [m], efficiency = 78%
Storage tank: storage capacity = 12,000 [m$^3$]

GA parameters:
Number of generations: 600
Population: 100
Time step: 15 min.
Probability of schedule mutation: 0.09 or 9 % / Population / Generation
Switches per chromosome: 2
Probability of S/C mutation: 9 % / Population / Generation
Magnification factor: 1000
Table A2-1: A comparison of the best daily cost at different generation numbers and using two different magnification exponents.

<table>
<thead>
<tr>
<th>Generation number</th>
<th>Magnification exponent $P = 20$</th>
<th>Magnification exponent $P = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best cost</td>
<td>Average cost</td>
</tr>
<tr>
<td>0</td>
<td>24000.38</td>
<td>151.03</td>
</tr>
<tr>
<td>1</td>
<td>85.90</td>
<td>82.64</td>
</tr>
<tr>
<td>2</td>
<td>85.37</td>
<td>82.64</td>
</tr>
<tr>
<td>3</td>
<td>83.19</td>
<td>82.64</td>
</tr>
<tr>
<td>4</td>
<td>83.05</td>
<td>82.41</td>
</tr>
<tr>
<td>5</td>
<td>82.96</td>
<td>81.73</td>
</tr>
<tr>
<td>6</td>
<td>82.89</td>
<td>81.73</td>
</tr>
<tr>
<td>7</td>
<td>83.03</td>
<td>78.32</td>
</tr>
<tr>
<td>8</td>
<td>82.68</td>
<td>77.53</td>
</tr>
<tr>
<td>9</td>
<td>82.52</td>
<td>75.83</td>
</tr>
<tr>
<td>10</td>
<td>82.07</td>
<td>75.83</td>
</tr>
<tr>
<td>11</td>
<td>81.80</td>
<td>75.83</td>
</tr>
<tr>
<td>12</td>
<td>81.38</td>
<td>75.83</td>
</tr>
<tr>
<td>13</td>
<td>81.19</td>
<td>75.83</td>
</tr>
<tr>
<td>14</td>
<td>80.37</td>
<td>75.83</td>
</tr>
<tr>
<td>15</td>
<td>79.24</td>
<td>78.61</td>
</tr>
<tr>
<td>16</td>
<td>78.87</td>
<td>78.46</td>
</tr>
<tr>
<td>17</td>
<td>78.42</td>
<td>80.81</td>
</tr>
<tr>
<td>18</td>
<td>78.15</td>
<td>78.06</td>
</tr>
<tr>
<td>19</td>
<td>77.86</td>
<td>78.38</td>
</tr>
<tr>
<td>20</td>
<td>77.75</td>
<td>77.86</td>
</tr>
<tr>
<td>25</td>
<td>75.83</td>
<td>75.83</td>
</tr>
<tr>
<td>250</td>
<td>72.42</td>
<td>76.32</td>
</tr>
<tr>
<td>300</td>
<td>67.59</td>
<td>73.35</td>
</tr>
<tr>
<td>350</td>
<td>67.23</td>
<td>71.15</td>
</tr>
<tr>
<td>400</td>
<td>62.20</td>
<td>71.76</td>
</tr>
<tr>
<td>450</td>
<td>58.79</td>
<td>60.23</td>
</tr>
<tr>
<td>500</td>
<td>55.38</td>
<td>58.09</td>
</tr>
<tr>
<td>550</td>
<td>55.38</td>
<td>55.88</td>
</tr>
<tr>
<td>600</td>
<td>55.38</td>
<td>55.95</td>
</tr>
</tbody>
</table>
Appendix 3 - Generating an approximate equation for the capital cost of storage reservoirs

The figure below shows the cost data points gathered from Muir, 1991. The data was used in a linear regression procedure which resulted in the following equation

\[
\text{Cost} [\$] = 305,691 + 142.33 \times \text{Capacity} [\text{m}^3]. \quad R^2 = 0.9819
\]

**Figure A3-1:** Capital cost vs. reservoir capacity

The above equation was reduced to the form shown below

\[
\text{Cost} [\$] = 300,000 + 140 \times \text{Capacity} [\text{m}^3]
\]
Table A3-1: Cost data from Muir, 1991

<table>
<thead>
<tr>
<th>Reservoir Capacity [cubic meters]</th>
<th>Total Cost [$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>340000</td>
</tr>
<tr>
<td>800</td>
<td>470000</td>
</tr>
<tr>
<td>1400</td>
<td>500000</td>
</tr>
<tr>
<td>2300</td>
<td>640000</td>
</tr>
<tr>
<td>2700</td>
<td>680000</td>
</tr>
<tr>
<td>3400</td>
<td>770000</td>
</tr>
<tr>
<td>4500</td>
<td>960000</td>
</tr>
</tbody>
</table>
Appendix 4 - Effect of the mutation exponents and mutation parameters on the average $G_p$ value and its respective standard deviation

Table below A4-1 through A4-3 were generated by shown hydraulic system and GA parameters that were used as input data for GAHOS-1B. The electricity cost data and water demand rates were provided by the input file 15-bit.txt shown in Appendix 14.

**Hydraulic System Parameters:**

Pump: flow = 0.191 [m$^3$/s], head = 60.3 [m], efficiency = 78%

Storage tank: storage capacity = 12,000 [m$^3$]

**GA parameters:**

Number of generations: 10,000, the GA was stopped prior to this if the $G_p$ was reached.

Population: 100

Time step: 15 min.

Probability of mutation: an experimental variable, look at the tables

Switches per chromosome: 2

**Note:** Each “run” shown in the tables is a complete execution of the GA till completion. Please don’t confuse a run with a generation, each run can simulate up to 10,000 generations.
Table A4-1: Gp values for various Magnification exponents ($P$), mutation Parameters equal to 0.09

<table>
<thead>
<tr>
<th>Runs \ $P$</th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>3</th>
<th>2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>183</td>
<td>193</td>
<td>170</td>
<td>182</td>
<td>447</td>
<td>1496</td>
</tr>
<tr>
<td>2</td>
<td>233</td>
<td>233</td>
<td>147</td>
<td>216</td>
<td>662</td>
<td>1034</td>
</tr>
<tr>
<td>3</td>
<td>125</td>
<td>177</td>
<td>118</td>
<td>222</td>
<td>361</td>
<td>1019</td>
</tr>
<tr>
<td>4</td>
<td>175</td>
<td>272</td>
<td>167</td>
<td>331</td>
<td>331</td>
<td>920</td>
</tr>
<tr>
<td>5</td>
<td>237</td>
<td>206</td>
<td>139</td>
<td>268</td>
<td>337</td>
<td>1287</td>
</tr>
<tr>
<td>6</td>
<td>121</td>
<td>144</td>
<td>234</td>
<td>294</td>
<td>422</td>
<td>1124</td>
</tr>
<tr>
<td>7</td>
<td>195</td>
<td>176</td>
<td>309</td>
<td>542</td>
<td>1359</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>110</td>
<td>207</td>
<td>172</td>
<td>315</td>
<td>360</td>
<td>1210</td>
</tr>
<tr>
<td>9</td>
<td>150</td>
<td>191</td>
<td>169</td>
<td>325</td>
<td>519</td>
<td>1603</td>
</tr>
<tr>
<td>10</td>
<td>390</td>
<td>156</td>
<td>254</td>
<td>260</td>
<td>373</td>
<td>448</td>
</tr>
<tr>
<td>11</td>
<td>230</td>
<td>138</td>
<td>219</td>
<td>230</td>
<td>667</td>
<td>1630</td>
</tr>
<tr>
<td>12</td>
<td>155</td>
<td>220</td>
<td>229</td>
<td>277</td>
<td>455</td>
<td>1093</td>
</tr>
<tr>
<td>13</td>
<td>279</td>
<td>192</td>
<td>164</td>
<td>258</td>
<td>576</td>
<td>518</td>
</tr>
<tr>
<td>14</td>
<td>189</td>
<td>220</td>
<td>172</td>
<td>294</td>
<td>599</td>
<td>811</td>
</tr>
<tr>
<td>15</td>
<td>203</td>
<td>285</td>
<td>153</td>
<td>285</td>
<td>289</td>
<td>868</td>
</tr>
<tr>
<td>16</td>
<td>218</td>
<td>323</td>
<td>228</td>
<td>284</td>
<td>471</td>
<td>809</td>
</tr>
<tr>
<td>17</td>
<td>137</td>
<td>172</td>
<td>223</td>
<td>271</td>
<td>424</td>
<td>638</td>
</tr>
<tr>
<td>18</td>
<td>209</td>
<td>228</td>
<td>194</td>
<td>316</td>
<td>812</td>
<td>1505</td>
</tr>
<tr>
<td>19</td>
<td>138</td>
<td>150</td>
<td>243</td>
<td>205</td>
<td>560</td>
<td>3614</td>
</tr>
<tr>
<td>20</td>
<td>241</td>
<td>173</td>
<td>169</td>
<td>217</td>
<td>359</td>
<td>1100</td>
</tr>
<tr>
<td>21</td>
<td>228</td>
<td>258</td>
<td>250</td>
<td>336</td>
<td>545</td>
<td>948</td>
</tr>
<tr>
<td>22</td>
<td>154</td>
<td>120</td>
<td>222</td>
<td>294</td>
<td>367</td>
<td>2982</td>
</tr>
<tr>
<td>23</td>
<td>212</td>
<td>156</td>
<td>289</td>
<td>250</td>
<td>548</td>
<td>813</td>
</tr>
<tr>
<td>24</td>
<td>193</td>
<td>162</td>
<td>182</td>
<td>255</td>
<td>926</td>
<td>1610</td>
</tr>
<tr>
<td>25</td>
<td>203</td>
<td>230</td>
<td>348</td>
<td>294</td>
<td>821</td>
<td>830</td>
</tr>
<tr>
<td><strong>Average Gp</strong></td>
<td>190</td>
<td>195</td>
<td>194</td>
<td>261</td>
<td>491</td>
<td>1203</td>
</tr>
<tr>
<td><strong>S.D. of Gp</strong></td>
<td>67</td>
<td>62</td>
<td>63</td>
<td>66</td>
<td>191</td>
<td>733</td>
</tr>
</tbody>
</table>

A summary of the above table.

<table>
<thead>
<tr>
<th>Magnification Exponent ($P$)</th>
<th>Gp Average</th>
<th>Standard Deviation of Gp</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>1203</td>
<td>733</td>
</tr>
<tr>
<td>3</td>
<td>491</td>
<td>191</td>
</tr>
<tr>
<td>5</td>
<td>261</td>
<td>66</td>
</tr>
<tr>
<td>10</td>
<td>194</td>
<td>63</td>
</tr>
<tr>
<td>15</td>
<td>195</td>
<td>62</td>
</tr>
<tr>
<td>20</td>
<td>190</td>
<td>67</td>
</tr>
</tbody>
</table>
Table A4-2: Gp values for various Mutation values when P=15

<table>
<thead>
<tr>
<th>Runs</th>
<th>Mutation</th>
<th>0.03</th>
<th>0.06</th>
<th>0.09</th>
<th>0.12</th>
<th>0.15</th>
<th>0.25</th>
<th>0.35</th>
<th>0.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>926</td>
<td>241</td>
<td>193</td>
<td>106</td>
<td>112</td>
<td>135</td>
<td>74</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1191</td>
<td>296</td>
<td>233</td>
<td>110</td>
<td>107</td>
<td>75</td>
<td>90</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>645</td>
<td>300</td>
<td>177</td>
<td>149</td>
<td>97</td>
<td>117</td>
<td>101</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>569</td>
<td>393</td>
<td>272</td>
<td>169</td>
<td>140</td>
<td>104</td>
<td>104</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>285</td>
<td>255</td>
<td>206</td>
<td>257</td>
<td>117</td>
<td>90</td>
<td>60</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>518</td>
<td>263</td>
<td>144</td>
<td>189</td>
<td>125</td>
<td>83</td>
<td>73</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1088</td>
<td>246</td>
<td>260</td>
<td>216</td>
<td>200</td>
<td>115</td>
<td>81</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>664</td>
<td>235</td>
<td>207</td>
<td>123</td>
<td>99</td>
<td>100</td>
<td>90</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1542</td>
<td>394</td>
<td>191</td>
<td>143</td>
<td>213</td>
<td>104</td>
<td>77</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1212</td>
<td>382</td>
<td>156</td>
<td>141</td>
<td>92</td>
<td>74</td>
<td>80</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>610</td>
<td>384</td>
<td>138</td>
<td>216</td>
<td>132</td>
<td>97</td>
<td>128</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>570</td>
<td>178</td>
<td>220</td>
<td>107</td>
<td>149</td>
<td>116</td>
<td>82</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1049</td>
<td>349</td>
<td>192</td>
<td>157</td>
<td>165</td>
<td>78</td>
<td>72</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>717</td>
<td>521</td>
<td>220</td>
<td>182</td>
<td>126</td>
<td>111</td>
<td>68</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>775</td>
<td>295</td>
<td>285</td>
<td>137</td>
<td>118</td>
<td>71</td>
<td>69</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>750</td>
<td>196</td>
<td>323</td>
<td>105</td>
<td>109</td>
<td>87</td>
<td>68</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1817</td>
<td>319</td>
<td>172</td>
<td>141</td>
<td>117</td>
<td>78</td>
<td>86</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1181</td>
<td>293</td>
<td>228</td>
<td>156</td>
<td>105</td>
<td>85</td>
<td>83</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>740</td>
<td>545</td>
<td>150</td>
<td>150</td>
<td>129</td>
<td>73</td>
<td>72</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>759</td>
<td>382</td>
<td>173</td>
<td>172</td>
<td>151</td>
<td>96</td>
<td>113</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1199</td>
<td>576</td>
<td>258</td>
<td>130</td>
<td>99</td>
<td>81</td>
<td>65</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>743</td>
<td>335</td>
<td>120</td>
<td>161</td>
<td>99</td>
<td>87</td>
<td>98</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>891</td>
<td>523</td>
<td>156</td>
<td>120</td>
<td>143</td>
<td>75</td>
<td>79</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>854</td>
<td>233</td>
<td>162</td>
<td>144</td>
<td>123</td>
<td>134</td>
<td>113</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>501</td>
<td>278</td>
<td>230</td>
<td>208</td>
<td>165</td>
<td>64</td>
<td>60</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>Average Gp</td>
<td>872</td>
<td>336</td>
<td>195</td>
<td>156</td>
<td>129</td>
<td>93</td>
<td>83</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>S.D. of Gp</td>
<td>347</td>
<td>110</td>
<td>62</td>
<td>39</td>
<td>31</td>
<td>20</td>
<td>18</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

A summary of the above table.

<table>
<thead>
<tr>
<th>Mutation Parameters</th>
<th>Gp Average</th>
<th>Standard Deviation of Gp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>872</td>
<td>347</td>
</tr>
<tr>
<td>0.06</td>
<td>336</td>
<td>110</td>
</tr>
<tr>
<td>0.09</td>
<td>195</td>
<td>62</td>
</tr>
<tr>
<td>0.12</td>
<td>156</td>
<td>39</td>
</tr>
<tr>
<td>0.15</td>
<td>129</td>
<td>31</td>
</tr>
<tr>
<td>0.25</td>
<td>93</td>
<td>20</td>
</tr>
<tr>
<td>0.35</td>
<td>83</td>
<td>18</td>
</tr>
<tr>
<td>0.50</td>
<td>80</td>
<td>19</td>
</tr>
</tbody>
</table>
Table A4-3: Gp values for various Mutation values when P=10

<table>
<thead>
<tr>
<th>Runs \ Mutation</th>
<th>0.09</th>
<th>0.12</th>
<th>0.18</th>
<th>0.25</th>
<th>0.30</th>
<th>0.35</th>
<th>0.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>170</td>
<td>202</td>
<td>190</td>
<td>120</td>
<td>91</td>
<td>112</td>
<td>112</td>
</tr>
<tr>
<td>2</td>
<td>147</td>
<td>140</td>
<td>200</td>
<td>104</td>
<td>68</td>
<td>112</td>
<td>112</td>
</tr>
<tr>
<td>3</td>
<td>118</td>
<td>155</td>
<td>130</td>
<td>86</td>
<td>146</td>
<td>90</td>
<td>104</td>
</tr>
<tr>
<td>4</td>
<td>167</td>
<td>117</td>
<td>129</td>
<td>113</td>
<td>147</td>
<td>139</td>
<td>186</td>
</tr>
<tr>
<td>5</td>
<td>139</td>
<td>126</td>
<td>140</td>
<td>128</td>
<td>163</td>
<td>131</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>176</td>
<td>123</td>
<td>110</td>
<td>106</td>
<td>94</td>
<td>100</td>
<td>413</td>
</tr>
<tr>
<td>7</td>
<td>176</td>
<td>199</td>
<td>110</td>
<td>134</td>
<td>106</td>
<td>274</td>
<td>238</td>
</tr>
<tr>
<td>8</td>
<td>169</td>
<td>245</td>
<td>111</td>
<td>125</td>
<td>113</td>
<td>124</td>
<td>182</td>
</tr>
<tr>
<td>9</td>
<td>254</td>
<td>141</td>
<td>142</td>
<td>111</td>
<td>99</td>
<td>153</td>
<td>182</td>
</tr>
<tr>
<td>10</td>
<td>219</td>
<td>142</td>
<td>134</td>
<td>119</td>
<td>113</td>
<td>249</td>
<td>471</td>
</tr>
<tr>
<td>11</td>
<td>229</td>
<td>116</td>
<td>148</td>
<td>143</td>
<td>115</td>
<td>147</td>
<td>184</td>
</tr>
<tr>
<td>12</td>
<td>164</td>
<td>126</td>
<td>135</td>
<td>96</td>
<td>151</td>
<td>130</td>
<td>120</td>
</tr>
<tr>
<td>13</td>
<td>172</td>
<td>147</td>
<td>108</td>
<td>126</td>
<td>111</td>
<td>135</td>
<td>347</td>
</tr>
<tr>
<td>14</td>
<td>153</td>
<td>190</td>
<td>145</td>
<td>122</td>
<td>91</td>
<td>214</td>
<td>96</td>
</tr>
<tr>
<td>15</td>
<td>228</td>
<td>165</td>
<td>115</td>
<td>104</td>
<td>110</td>
<td>175</td>
<td>174</td>
</tr>
<tr>
<td>16</td>
<td>223</td>
<td>168</td>
<td>115</td>
<td>98</td>
<td>124</td>
<td>97</td>
<td>289</td>
</tr>
<tr>
<td>17</td>
<td>194</td>
<td>181</td>
<td>138</td>
<td>102</td>
<td>77</td>
<td>123</td>
<td>223</td>
</tr>
<tr>
<td>18</td>
<td>243</td>
<td>120</td>
<td>119</td>
<td>102</td>
<td>120</td>
<td>115</td>
<td>144</td>
</tr>
<tr>
<td>19</td>
<td>169</td>
<td>204</td>
<td>93</td>
<td>79</td>
<td>111</td>
<td>93</td>
<td>213</td>
</tr>
<tr>
<td>20</td>
<td>250</td>
<td>176</td>
<td>117</td>
<td>88</td>
<td>105</td>
<td>119</td>
<td>152</td>
</tr>
<tr>
<td>21</td>
<td>222</td>
<td>142</td>
<td>130</td>
<td>117</td>
<td>100</td>
<td>73</td>
<td>196</td>
</tr>
<tr>
<td>22</td>
<td>289</td>
<td>136</td>
<td>106</td>
<td>95</td>
<td>184</td>
<td>116</td>
<td>299</td>
</tr>
<tr>
<td>23</td>
<td>182</td>
<td>121</td>
<td>105</td>
<td>119</td>
<td>100</td>
<td>118</td>
<td>401</td>
</tr>
<tr>
<td>24</td>
<td>348</td>
<td>166</td>
<td>115</td>
<td>125</td>
<td>94</td>
<td>124</td>
<td>271</td>
</tr>
<tr>
<td>25</td>
<td>194</td>
<td>158</td>
<td>125</td>
<td>111</td>
<td>115</td>
<td>136</td>
<td>224</td>
</tr>
</tbody>
</table>

| Average Gp     | 194  | 158  | 125  | 111  | 115  | 136  | 224  |
| S.D. of Gp      | 63   | 34   | 22   | 17   | 26   | 48   | 102  |

A summary of the above table.

<table>
<thead>
<tr>
<th>Mutation Parameters</th>
<th>Gp Average</th>
<th>Gp Average</th>
<th>Standard Deviation of Gp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.09</td>
<td>194</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>0.12</td>
<td>158</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>0.18</td>
<td>125</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>111</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td>115</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>0.35</td>
<td>136</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>224</td>
<td>102</td>
<td></td>
</tr>
</tbody>
</table>
Appendix 5  Tables concerning the model validation (third stage)

Table A5-1:  Total System Cost for all possible combinations

<table>
<thead>
<tr>
<th>Combination</th>
<th>Electricity Cost [$/day]</th>
<th>Total System Cost [$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank 1 - Pump 1</td>
<td>77.53</td>
<td>3,582,489</td>
</tr>
<tr>
<td>Tank 1 - Pump 2</td>
<td>66.85</td>
<td>3,673,742</td>
</tr>
<tr>
<td>Tank 1 - Pump 3</td>
<td>81.44</td>
<td>3,601,887</td>
</tr>
<tr>
<td>Tank 1 - Pump 4</td>
<td>73.42</td>
<td>3,545,599</td>
</tr>
<tr>
<td>Tank 1 - Pump 5</td>
<td>85.99</td>
<td>3,898,475</td>
</tr>
<tr>
<td>Tank 2 - Pump 1</td>
<td>58.79</td>
<td>3,986,837</td>
</tr>
<tr>
<td>Tank 2 - Pump 2</td>
<td>51.45</td>
<td>4,090,859</td>
</tr>
<tr>
<td>Tank 2 - Pump 3</td>
<td>60.66</td>
<td>3,998,460</td>
</tr>
<tr>
<td>Tank 2 - Pump 4</td>
<td>54.02</td>
<td>3,947,456</td>
</tr>
<tr>
<td>Tank 2 - Pump 5</td>
<td>64.51</td>
<td>4,292,391</td>
</tr>
<tr>
<td>Tank 3 - Pump 1</td>
<td>53.68</td>
<td>4,676,817</td>
</tr>
<tr>
<td>Tank 3 - Pump 2</td>
<td>51.64</td>
<td>4,801,068</td>
</tr>
<tr>
<td>Tank 3 - Pump 3</td>
<td>53.10</td>
<td>4,679,113</td>
</tr>
<tr>
<td>Tank 3 - Pump 4</td>
<td>42.39</td>
<td>4,612,544</td>
</tr>
<tr>
<td>Tank 3 - Pump 5</td>
<td>58.65</td>
<td>4,979,526</td>
</tr>
<tr>
<td>Tank 4 - Pump 1</td>
<td>53.68</td>
<td>5,649,377</td>
</tr>
<tr>
<td>Tank 4 - Pump 2</td>
<td>51.64</td>
<td>5,977,819</td>
</tr>
<tr>
<td>Tank 4 - Pump 3</td>
<td>51.21</td>
<td>5,848,657</td>
</tr>
<tr>
<td>Tank 4 - Pump 4</td>
<td>42.39</td>
<td>5,789,294</td>
</tr>
<tr>
<td>Tank 4 - Pump 5</td>
<td>58.65</td>
<td>6,156,276</td>
</tr>
<tr>
<td>Tank 5 - Pump 1</td>
<td>53.68</td>
<td>7,252,250</td>
</tr>
<tr>
<td>Tank 5 - Pump 2</td>
<td>51.64</td>
<td>7,376,501</td>
</tr>
<tr>
<td>Tank 5 - Pump 3</td>
<td>51.21</td>
<td>7,247,339</td>
</tr>
<tr>
<td>Tank 5 - Pump 4</td>
<td>42.39</td>
<td>7,187,976</td>
</tr>
<tr>
<td>Tank 5 - Pump 5</td>
<td>58.65</td>
<td>7,554,959</td>
</tr>
</tbody>
</table>
Table A5-2:  Cost properties for five Storage tanks of varying sizes

<table>
<thead>
<tr>
<th>Designation</th>
<th>Storage size [m³]</th>
<th>Principal Cost [$]</th>
<th>M &amp; O Cost [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank 1</td>
<td>2000</td>
<td>580000</td>
<td>7.0</td>
</tr>
<tr>
<td>Tank 2</td>
<td>4000</td>
<td>860000</td>
<td>6.9</td>
</tr>
<tr>
<td>Tank 3</td>
<td>7000</td>
<td>1280000</td>
<td>6.8</td>
</tr>
<tr>
<td>Tank 4</td>
<td>12000</td>
<td>1980000</td>
<td>6.7</td>
</tr>
<tr>
<td>Tank 5</td>
<td>18000</td>
<td>2820000</td>
<td>6.6</td>
</tr>
</tbody>
</table>

Table A5-3:  Cost properties for five different pumps

<table>
<thead>
<tr>
<th>Designation</th>
<th>Principal Cost [$]</th>
<th>M &amp; O Cost [%]</th>
<th>Flow [m³/s]</th>
<th>Head [m]</th>
<th>Efficiency [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pump 1</td>
<td>1,116,090</td>
<td>10</td>
<td>0.191</td>
<td>60.3</td>
<td>78</td>
</tr>
<tr>
<td>Pump 2</td>
<td>1,023,706</td>
<td>13</td>
<td>0.180</td>
<td>53.9</td>
<td>81</td>
</tr>
<tr>
<td>Pump 3</td>
<td>1,178,509</td>
<td>9</td>
<td>0.203</td>
<td>62.1</td>
<td>77</td>
</tr>
<tr>
<td>Pump 4</td>
<td>1,231,594</td>
<td>8</td>
<td>0.231</td>
<td>55.3</td>
<td>76</td>
</tr>
<tr>
<td>Pump 5</td>
<td>1,193,829</td>
<td>11</td>
<td>0.196</td>
<td>68.2</td>
<td>79</td>
</tr>
</tbody>
</table>
Appendix 6 - Average day water demand rates (AVRDAY 1)

<table>
<thead>
<tr>
<th>Time Period (hr)</th>
<th>Water Demand [m³/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.080</td>
</tr>
<tr>
<td>2</td>
<td>0.075</td>
</tr>
<tr>
<td>3</td>
<td>0.065</td>
</tr>
<tr>
<td>4</td>
<td>0.062</td>
</tr>
<tr>
<td>5</td>
<td>0.065</td>
</tr>
<tr>
<td>6</td>
<td>0.065</td>
</tr>
<tr>
<td>7</td>
<td>0.066</td>
</tr>
<tr>
<td>8</td>
<td>0.091</td>
</tr>
<tr>
<td>9</td>
<td>0.102</td>
</tr>
<tr>
<td>10</td>
<td>0.101</td>
</tr>
<tr>
<td>11</td>
<td>0.115</td>
</tr>
<tr>
<td>12</td>
<td>0.120</td>
</tr>
<tr>
<td>13</td>
<td>0.130</td>
</tr>
<tr>
<td>14</td>
<td>0.140</td>
</tr>
<tr>
<td>15</td>
<td>0.135</td>
</tr>
<tr>
<td>16</td>
<td>0.130</td>
</tr>
<tr>
<td>17</td>
<td>0.128</td>
</tr>
<tr>
<td>18</td>
<td>0.132</td>
</tr>
<tr>
<td>19</td>
<td>0.140</td>
</tr>
<tr>
<td>20</td>
<td>0.160</td>
</tr>
<tr>
<td>21</td>
<td>0.162</td>
</tr>
<tr>
<td>22</td>
<td>0.161</td>
</tr>
<tr>
<td>23</td>
<td>0.150</td>
</tr>
<tr>
<td>24</td>
<td>0.135</td>
</tr>
</tbody>
</table>
Appendix 7 - Description of various functions used in GAHOS

dis_rate
A float representing the discount rate of money. It is in fractional form not in the form of a percentage. It is utilized in the P_given_A equation.

inflation
A float representing the annual rate of inflation. It is in fractional form not in the form of a percentage. It is utilized in the P_given_A equation.

length
A float representing the length of the project in years. It is utilized in the P_given_A equation.

ngen
An integer representing the number of generations GAHOS must simulate. There is no restriction on the magnitude of ngen.

num_of_pumps
An integer representing the number of pumps in the inventory (i.e. that could possibly be selected. This number must be kept below 100 (see pumps).

num_of_tanks
An integer representing the number of tanks in the inventory (i.e. that could possibly be selected. This number must be kept below 100 (see tanks).

P_given_A
A float representing the present worth factor all the annual cost over the length of a project. Their are three input variables inflation, length and dis_rate.

price
A 96 long array of floats. It represents the electricity cost [$/kW·hr] for a 24 hour period in 15 min. intervals. The data is collected in the opening_file subroutine and it mainly used in the

pumps
A 100x5 array of floats. It is used to contain all pump parameters and can easily be increased in size. It is initially used in the opening_file subroutine. The first parameter is capital cost [$], second is O&M cost as a fraction of the capital cost and the third forth and fifth parameters are discharge [m$^3$/s], head [m] and efficiency respectively.
size
An integer representing the number of chromosome in a population. It is restricted to 100 chromosomes or less. This may be increased by redefining the magnitude of the fixed arrays involved.

tanks
A 100x3 array of floats. It is used to contain all reservoir parameters and can easily be increased in size. It is initially used in the opening file subroutine. The first parameter is capital cost [$], second is O&M cost as a fraction of the capital cost and the third parameter is capacity [m$^3$].
Appendix 8 - GAHOS-1B Computer Code

#include <conio.h>
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <time.h>
#include <dos.h>
#include <math.h>
#include <stdlib.h>
#include <values.h>

char ifile[10]; /*global variables*/
int ansr;
int size; /*global variables*/
int ngen; /*global variables*/
int time_minutes;
int time_max;
float prob;
int changes;
float abs_best;
float demand[96]; /*global variables*/
float price[96]; /*global variables*/
int orig_chromo[100][96];
int test_chromo[96];
float evaluate[100];
float storage_remaining[97];
int list[100];
int new_chromo_list[100][96];
int order_of_evaluation[100];
float storage_specs[100][2];
float new_storage_specs[100][2];
float running_total[100];
float gen_best[1000];
float gen_average[1000];

void opening_file(void); /*function prototype*/
void output_file(void);
void screen_data(void); /*function prototype*/
void title(void); /*function prototype*/
void run_time(void);
void rand_list(void);
void print_list(void);
void crossover(int a);
void show_crossover_title(void);
void show_crossover(void);
void show_pair(void);
void show_mutation(void);
void show_evaluation(void);
void show_done(void);
void show_generation(int a);
void roulette_wheel(void);
void mutation(float x);
void qsort(float v[], int left, int right);

void swap(float v[], int i, int j);
void swap2(int v[], int i, int j);

void expand_price(void);
void expand_demand(void);
float final_storage(int i);
int u_rand(int a);
float unit(void);
int random(int num);
void eval(void);
void children(void);
float storage_max;
float storage_mut;
float Q;
float H;
float n;
float Po; // magnification exponent/
float gen_total;
int best=99;
float high=-999;

clock_t start, end;

void main(void) /* main program*/
{

    int i,j,p,ans,some_number;
    randomize();
    start = clock();
title();
screen_data();
title();
opening_file();
expand_price();
expand_demand();
eval();
gen_total=0;
abs_best=1000;

for (j=0;j<=size-1;j++){
    gen_total=gen_total + evaluate[j];
}

gen_average[0] = gen_total/size;
qsort(evaluate,0,99);
gen_best[0]=evaluate[0];
ansr=1;

for (p=1;p<=ngen;p++){
    show_generation(p);
    rand_list();
    show_pair();
    if (p<ngen){
        for (i=0;i<=size-2;i=i+2){
            crossover(i);
            show_crossover();
        }
        show_mutation();
        show_evaluation();
        eval();
        roulette_wheel();
        gotoxy(10,13);
        eval();
        gen_total=0;
        qsort(evaluate,0,99);
        if (abs_best>evaluate[0]){
            abs_best=evaluate[0];
            some_number=random(size);
            for (i=0;i<=time_max-1;i++){
                orig_chromo[some_number][i]=orig_chromo[order_of_evaluation[0]][i];
            }
            for (i=0;i<=2;i++){
                storage_specs[some_number][i]=storage_specs[order_of_evaluation[0]][i];
            }
        }
    }
}

98
if (p==ngen)
{
    show_done();
    gotoxy(10,12);
    printf("time elapsed(sec): ");
    run_time();
    eval();
    qsort(evaluate,0,99);
    gotoxy(10,16);
}

output_file();

void title(void)
{
    clrscr();
    gotoxy(15,1);
    printf("GAHOS-1 By: Tobias A. Bloch (With absolute-best function)");
}

void output_file(void)
{
    int i,j;
    FILE *stream2;
    stream2 = fopen("wbest.txt","w");
    for (i=0;i<=time_max-1;i++)
    {
        fprintf(stream2,"\n%d %f %f",i,price[i],demand[i]);
    }
}
fpnrf(stream2,"\nThe best solution\n\n");

for (i=0;i<=ngen-1;i++){
    fprnrf(stream2,"\n%d\t%f",i,gen_best[i]);
}

void opening_file(void)
{
    int i,j,temp_a,temp_b;
    FILE *stream1;
    stream1 = fopen(ifile,"r");
    gotoxy(10,5);
    printf("Opening file: %s",ifile);
    gotoxy(10,6); /*start location of line below */
    printf("Reading file: ");

    temp_a=0;

    for (i=0;i<=23;i++){
        fscanf(stream1,"%f",&price[i]);
        gotoxy(24,6); /*pretend info*/
        printf("%d",demand[i]);
    }

    for (i=0;i<=23;i++){
        fscanf(stream1,"%f",&demand[i]);
        gotoxy(24,6);
        printf("%f",price[i]);
    }

    gotoxy(10,6);
    printf("chromosomes: ");

    for (j=0;j<=size-1;j++){
        fscanf(stream1,"\n");
        gotoxy(23,6);
        for (i=0;i<=time_max-1;i++){
            temp_b=pow(-1,i*j);
            orig_chromo[j][i]=max(temp_a,temp_b); //random(2);
            storage_specs[j][1]=pow((j/(size-1)),0.2); //random(storage_max+1)/storage_max;
            if (time_minutes>15){
void screen_data(void)
{
    gotoxy(1,4);
    printf("Enter input file name: ");
    gets(ifile);
    gotoxy(1,5);
    printf("Enter the time steps (minutes): ");
    //time_minutes=15;
    scanf("%d",&time_minutes);
    gotoxy(1,6);
    printf("Enter the population size: ");
    //size=100;
    scanf("%d",&size);
    gotoxy(1,7);
    printf("Enter the number of generations: ");
    scanf("%d",&ngen);
    gotoxy(1,8);
    printf("Enter the probability of mutation: ");
    //prob=0.09;
    scanf("%f",&prob);
    gotoxy(1,9);
    printf("Enter the mutation per chromosome: ");
    //changes=2;
    scanf("%d",&changes);
    gotoxy(1,10);
    printf("Enter the storage capacity: ");
    //storage_max=12000;
    scanf("%f",&storage_max);
    gotoxy(1,11);
}

printf("%d",orig_chromo[j][i]);

}
printf("Enter the storage mutation: ");
//storage_mut=0.09;
scanf("%f",&storage_mut);
gotoxy(1,12);
printf("Enter the pump flow: ");
//Q=0.191;
scanf("%f",&Q);
gotoxy(1,13);
printf("Enter the pump head: ");
//H=60.3;
scanf("%f",&H);
gotoxy(1,14);
printf("Enter the pump efficiency: ");
//n=0.78;
scanf("%f",&n);
gotoxy(1,15);
printf("Enter the magnification Exponent: ");
scanf("%f",&Po);
time_max=(60/time_minutes)*24;
}

void run_time(void)
{
    end = clock();
    printf("%.1f", (end - start) / CLK_TCK);
}

void rand_list(void)
{
    int i, loca_1, loca_2, temp_of_loca_1;

    for (i=0;i<=size-1;i++)
    {
        list[i]=i;
    }

    for (i=1;i<=4*size;i++)
    { 
        loca_1=u_rand(size);
        loca_2=u_rand(size);
        temp_of_loca_1=list[loca_1];
        list[loca_1]=list[loca_2];
        list[loca_2]=temp_of_loca_1;
    }
}
void crossover(int a)
{
    int temp;
    float temp2;
    int pos,i;
    float average_ratio;
    pos=random(time_max-2);

    //the everaging of the ratios
    average_ratio=(storage_specs[list[a]][1]+storage_specs[list[a+1]][1])/2;
    storage_specs[list[a]][1]=average_ratio;
    storage_specs[list[a+1]][1]=average_ratio;

    for (i=pos;i<=time_max-1;i++){
        temp=orig_chromo[list[a]][i];
        orig_chromo[list[a]][i]=orig_chromo[list[a+1]][i];
        orig_chromo[list[a+1]][i]=temp;
    }
}

void show_done(void)
{
    gotoxy(45,10);
    printf("done ");
}

void show_pair(void)
{
    gotoxy(45,10);
    printf("pairing ");
}

void show_mutation(void)
{
    gotoxy(45,10);
    printf("mutating ");
}

void show_evaluation(void)
{
void show_crossover(void)
{
    gotoxy(45,10);
    printf("crossover ");
}

void show_generation(int a)
{
    gotoxy(25,10);
    printf("%d ",a);
}

void expand_price(void)
{
    int i,j,c;
    c=24;
    if (time_max>24)
    {
        for (i=time_max-1;i>=0;i=i-(60/time_minutes))
        {
            c=c-1;
            for(j=0;j<(60/time_minutes);++j)
            {
                price[i-j]=price[c];
            }
        }
    }
}

void expand_demand(void)
{
    int i,j,c;
    c=24;
    if (time_max>24)
    {
        for (i=time_max-1;i>=0;i=i-(60/time_minutes))
        {
            c=c-1;
            for(j=0;j<(60/time_minutes);++j)
            {
                demand[i-j]=demand[c];
            }
        }
    }
}

float final_storage(int i)
```c
{
    int j;
    storage_remaining[0]=storage_specs[i][1]*storage_max;
    for (j=1;j<=time_max;j++){
        storage_remaining[j]=storage_remaining[j-1] + (Q*orig_chromo[i][j-1] -
                demand[j-1])*(3600/(60/time_minutes));
        if (storage_remaining[j] > storage_max){
            storage_remaining[j] = storage_max;
        }
    }
    printf("\n%ld %f",j,storage_remaining[j]);
}

return storage_remaining[time_max];
}

void x_print(void)
{
    int j;
    for (j=0;j<=time_max-1;j++)
    {
        printf("%d",test_chromo[j]);
    }
}

void eval(void)
{
    int i,j;
    float x;
    for (i=0;i<=size-1;i++){
        order_of_evaluation[i]=i;
        evaluate[i]=0;
        x=0;
        storage_remaining[0]=storage_specs[i][1]*storage_max;
        x=orig_chromo[i][0]*price[0]*(H*Q*9.8)/(60/time_minutes);
        for (j=1;j<=time_max;j++){
            storage_remaining[j]=storage_remaining[j-1] +
                (Q*orig_chromo[i][j-1] - demand[j-1])*60*time_minutes;
            if (j<time_max)
            {
                x= x +
                orig_chromo[i][j]*price[j]*(H*Q*9.8)/(60/time_minutes);
            }
        }
```
```c
void print_list(void)
{
    int i;

    for (i=1;i<=size;i++){
        printf("%d %d",i,list[i]);
    }
}
```
void mutation(float x)
{
    int i,j,number_chromo,a,b,c,temp,t,number_storage_mut;
    number_chromo=(size*x)/1;
    number_storage_mut=(size*storage_mut)/1;

    for (i=0;i<=number_chromo;i++){

        for (j=1;j<=changes;j++){
            a=random(size);
            b=random(time_max); // (time_max);
            c=random(time_max); // (time_max);

            temp=orig_chromo[a][b];
            orig_chromo[a][b]=orig_chromo[a][c];
            orig_chromo[a][c]=temp;
        }

        for (i=0;i<=number_chromo;i++){
            a=random(size);
            b=random(time_max); // (time_max);

            if (orig_chromo[a][b]==1)
                orig_chromo[a][b]=0;
            else
                orig_chromo[a][b]=0;
        }

    for (i=0;i<=number_storage_mut;i++){
        c=random(size);
        storage_specs[c][1]=random(storage_max+1)/storage_max;
    }
}

void roulette_wheel(void) 
{
int i,j,k,counter_inside_while;
float random_number;
gotoxy(10,21);

running_total[0]=0;
random_number=0;

for (i=0;i<=size-1;i++)
    running_total[i]=0; //initiate

for (i=1;i<=size-1;i++)
    running_total[i]=running_total[i-1] + 1000/pow(evaluate[i],Po);

for (i=0;i<=size-1;i++)
    counter_inside_while=size-1;
    random_number = unit()*running_total[size-1];

    for(k=1;k<=size-2;k++)
        if (((random_number > running_total[k])&&random_number < running_total[k+1]))
            counter_inside_while=k+1;
    }

if (random_number<running_total[0]){
    counter_inside_while=0;
}

for (j=0;j<=time_max-1;j++)
    new_chromo_list[i][j]=orig_chromo[counter_inside_while][j];

```c
void qsort(float v[], int left, int right)
{
    int i, last, last2;
    if (left >= right)
        return;
    swap(v, left, (left + right) / 2);
    swap2(order_of_evaluation, left, (left + right) / 2);
    last = left;
    last2 = left;
    for (i = left + 1; i <= right; i++)
    {
        if (v[i] < v[left])
        {
            swap(v, ++last, i);
            swap2(order_of_evaluation, ++last2, i);
        }
    }
    swap(v, left, last);
    swap2(order_of_evaluation, left, last2);
    qsort(v, left, last - 1);
    qsort(v, last + 1, right);
}

void swap(float v[], int i, int j)
{
    float temp;
    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}
```
void swap2(int v[], int i, int j)  
{  
    int temp;  
    temp = v[i];  
    v[i] = v[j];  
    v[j] = temp;  
}

void children(void)  
{  
    int i, j;  

    //this is new  
    for (i=0;i<=9;i++)  
        for (j=0;j<=time_max-1;j++)  
        {  
            orig_chromo[order_of_evaluation[i+90]][j]=orig_chromo[order_of_evaluation[0]]
                [j];  
            storage_specs[order_of_evaluation[i+90]][1]=storage_specs[order_of_evaluation[0]][1];  
        }  
    //new ends here  
}
Appendix 9 - GAHOS-2 Computer Code

#include <conio.h>
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <time.h>
#include <dos.h>
#include <math.h>
#include <stdlib.h>
#include <values.h>

char ifile[10]; /*global variables*/
int ans;
int some_number;
int size; /*global variables*/
int ngen; /*global variables*/
int time_minutes;
int time_max;
float prob;
int changes;
int num_of_pumps;
float P;
int num_of_tanks;
float length;
float dis_rate;
float inflation;
float P_given_A;
float absal_best;
float demand[96];/*global variables*/
float price[96];/*global variables*/
int best_orig_chromo[96];/*global variables*/
int best_second_chromo[3];
int orig_chromo[100][96];
int second_chromo[100][3];
int test_chromo[99];
float evaluate[100];
float storage_remaining[97];
int list[100];
int new_chromo_list[100][96];
int second_chromo_list[100][3];
int order_of_evaluation[100];
//float storage_specs[100][2];
float new_storage_specs[100][2];
float running_total[100];
float tanks[100][3];
float pumps[100][5];

void opening_file(void); /*function prototype*/
void output_file(void);
void screen_data(void); /*function prototype*/
void title(void); /*function prototype*/
void run_time(void);
void rand_list(void);
void print_list(void);
void crossover(int a);
void show_crossover_title(void);
void show_crossover(void);
void show_pair(void);
void show_mutation(void);
void show_evaluation(void);
void show_done(void);
void show_generation(int a);
void roulette_wheel(void);
void mutation(float x);
void qsort(float v[], int left, int right);
void qsort2(float v[], int left, int right);
void swap(float v[], int i, int j);
void swap2(int v[], int i, int j);
void expand_price(void);
void expand_demand(void);
//float final_storage(int i);
int u_rand(int a);
float unit(void);
int random(int num);
void eval(void);
void children(void);
int x_addition(void);
//float x_eval(void);
//void x_test(void);
void x_initial(void);
void x_print(void);
float storage_max;
float storage_mut;
float Q;
float H;
float n;
int best=-99;
float high=-999;

clock_t start, end;
void main(void) /* main program*/
{
    int i,j,pjunk;
    randomize();
    absal_best=10000000;
    start = clock();
    title();
    screen_data();
    title();
    opening_file();
    //printf("n probability %f",prob);
    // getch();
    expand_price();
    //printf("n probability %f",prob);
    // getch();
    expand_demand();
    //printf("n probability %f",prob);
    // getch();

    for (p=0;p<=ngen;p++)
    {
        show_generation(p);
        rand_list();
        show_pair();
        if (p<ngen)
        {
            for (i=0;i<=size-2;i=i+2){
                crossover(i);
                show_crossover();
            }
            show_mutation();
            mutation(prob);
            show_evaluation();
        }
    eval();
    //qsort(evaluate,0,99);

    if (p<ngen){
        some_number=random(size);
        for (i=0;i<=time_max-1;i++){
orig_chromo[some_number][i]=best_orig_chromo[i];
}
for (i=0;i<=2;i++){
    second_chromo[some_number][i]=best_second_chromo[i];
}

//children();
roulette_wheel();
mutation(prob);
eval();
//qsort(evaluate,0,99);
}
qsort(evaluate,0,size-1);
gotoxy(10,14);
printf("Current best: %.0f %.0f ",evaluate[0],absal_best);
gotoxy(10,15);
printf("%d",second_chromo[order_of_evaluation[0]][0],second_chromo[order_of_evaluation[0]][1],second_chromo[order_of_evaluation[0]][2]);

if (absal_best>evaluate[0]){
    absal_best=evaluate[0];
    for (i=0;i<=time_max-1;i++){

best_orig_chromo[i]=orig_chromo[order_of_evaluation[0]][i];
    }
    for (i=0;i<=2;i++){

best_second_chromo[i]=second_chromo[order_of_evaluation[0]][i];
    }
}

junk=random(size);
for (i=0;i<=time_max-1;i++){
    orig_chromo[junk][i]=best_orig_chromo[i];
}
for (i=0;i<=2;i++){
    second_chromo[junk][i]=best_second_chromo[i];
}
eval();
gotoxy(10,15);

//printf("Current best: %.1f ",evaluate[0]);
if (p==ngen){
    show_done();
    gotoxy(10,12);
    printf("time elapsed (sec): ");
    run_time();
    /*for (i=0;i<=time_max-1;i++){
        printf("%d %f %f %f %f Wpumps Cil [O];pumps [I,pumps [I,pumps [I,il
    }*/
    //getch();
    /*for (i=0;i<=time_max-1;i++){

        printf("%n%.4f",orig_chromo[order_of_evaluation[j]][i]*price[i]*140.27);
    }*/
    //getch();
    //gotoxy(10,16);
    //printf("Reservoir stage\n");
    //for (j=0;j<=0;j++){
        //printf("%n%d %.1f %f

.f\n",j,evaluate[j],final_storage(order_of_evaluation[j]));
        /*printf("The best solution: ");
        gotoxy(10,17);
        for (i=0;i<=time_max-1;i++){
            //printf("%d",orig_chromo[order_of_evaluation[0]][i]);
        }*/
    }
}

//printf("%n %d %d",random(time_max),random(time_max));
output_file();

/*gotoxy(2,18);
for (i=0;i<=num_of_pumps-1;i++){
    printf("%f %f %f %f

.f\n",pumps[i][0],pumps[i][1],pumps[i][2],pumps[i][3],pumps[i][4]);
}*/
gotoxy(2,18);

//printf("why\n");
//printf(" %fn",tanks[0][0]);
//printf(" %fn",tanks[0][1]);

//printf("%f %fn",tanks[0][0],tanks[0][1]);
for (i=0;i<=num_of_tanks-1;i++){


```c
#include <stdio.h>

int main() {
    // Initialize variables
    // Logic for pump control
    // Display output
    return 0;
}
```


int i,j;
FILE *stream2;
stream2 = fopen("out.txt","w");
for (i=0;i<=time_max-1;i++)
    fprintf(stream2,"\n%d %f %f",i,price[i],demand[i]);
}
fprintf(stream2,"\nThe best solution\n\n");
}
for (i=0;i<=num_of_pumps-1;i++)
    fprintf(stream2,"%0f %.2f %.3f %.1f
%2fn",pumps[i][0],pumps[i][1],pumps[i][2],pumps[i][3],pumps[i][4]);
/*for (j=0;j<size-1;j++){
    fprintf(stream2,"\n %3f",storage_specs[order_of_evaluation[j]][1]);
    for (i=0;i<=time_max-1;i++)
        fprintf(stream2,"%d",orig_chromo[order_of_evaluation[j]][i]);
}
*/
}
void opening_file(void)
{
    int i,j,a,b;
    FILE *stream1;
    stream1 = fopen(ifile,"r");
gotoxy(10,5);
    printf("Opening file: %s",ifile);
gotoxy(10,6); /*start location of line below*/
    printf("Reading file: ");
for (i=0;i<=23;i++)
    {    
    fscanf(stream1,"%f\n",&price[i]);
    gotoxy(24,6); /*pretend info*/
    printf("%d",demand[i]);
    }
for (i=0;i<=23;i++)
    {    
    fscanf(stream1,"%f\n",&demand[i]);
    gotoxy(24,6);
    printf("%f",price[i]);
    }
    //fscanf(stream1,"%f %f",&tanks[0][0],&tanks[0][1]);
    fscanf(stream1,"\n");
}
for (i=0;i<=num_of_tanks-1;i++)
    fscanf(stream1,"%f%f%f%f
",&tanks[i][0],&tanks[i][1],&tanks[i][2]);
}

fscanf(stream1,"
");

for (i=0;i<=num_of_pumps-1;i++)
    fscanf(stream1,"%f%f%f%f
%f
",&pumps[i][0],&pumps[i][1],&pumps[i][2],&pumps[i][3],&pumps[i][4]);
}

gotoxy(10,6);
printf("chromosomes: ");

for (j=0;j<=size-1;j++)
    fscanf(stream1,"
");
gotoxy(23,6);
for (i=0;i<=time_max-1;i++)
    // fscanf(stream1,"%d",&orig_chromo[j][i]);
    orig_chromo[j][i]=random(2);
    //storage_specs[j][i]=random(storage_max+1)/storage_max;
    if (time_minutes>15)
        printf("%d",orig_chromo[j][i]);

second_chromo[j][0]=random(1001);
//a=unit0*num_of_tanks;
//b=unit0*num_of_pumps;
second_chromo[j][1]=random(num_of_tanks);
second_chromo[j][2]=random(num_of_pumps);
}

gotoxy(10,6);
printf("chromosomes initiation: done ");
gotoxy(10,10);
printf("generation #: ");
gotoxy(35,10);
printf("status: ");

gotoxy(2,18);
for (i=0;i<=25;i++)
    //printf("%d %d %d
",second_chromo[i][0],second_chromo[i][1],second_chromo[i][2]);
}
void screen_data(void)
{

gotoxy(1,4);
printf("Enter input file name: ");
gets(input);
gotoxy(1,5);
printf("Enter the time steps (minutes): ");
scanf("%d", &time_minutes);
gotoxy(1,6);
printf("Enter the population size: ");
scanf("%d", &size);
gotoxy(1,7);
printf("Enter the number of generations: ");
scanf("%d", &ngen);
gotoxy(1,8);
printf("Enter the probability of mutation: ");
scanf("%f", &prob);
gotoxy(1,9);
printf("Enter the mutation per chromosome: ");
scanf("%d", &changes);
//gotoxy(1,10);
//printf("Enter the storage capacity: ");
//scanf("%f", &storage_max);

gotoxy(1,11);
printf("Enter the storage mutation: ");
scanf("%f", &storage_mut);

gotoxy(1,12);
printf("Enter the number of pumps: ");
scanf("%d", &num_of_pumps);

gotoxy(1,13);
printf("Enter the number of tanks: ");
scanf("%d", &num_of_tanks);

gotoxy(1,14);
printf("Enter the inflation rate: ");
scanf("%f", &inflation);

gotoxy(1,15);
printf("Enter the discount rate: ");
scanf("%f", &dis_rate);

gotoxy(1,16);
printf("Enter the simulation length: ");
scanf("%f", &length);

gotoxy(1,17);
printf("Enter the Magnification exponent: ");
scanf("%f", &Po);
}
Q=0.191;
H=60.3;
n=0.78;

P\_{\text{given\_A}}=(\text{pow}((1+\text{dis\_rate})/(1+\text{inflation})),\text{length})^{-1}/((\text{dis\_rate}+\text{inflation})
\quad \times (\text{pow}((1+\text{dis\_rate})/(1+\text{inflation})),\text{length}));

// P\_{\text{given\_A}}=1;

time\_max=(60/time\_minutes)*24;

void x\_initial(void)
{
    int i;
    for (i=0;i<=time\_max-1;i++)
    {
        test\_chromo[i]=0;
    }
}

int x\_addition(void)
{
    int i;
    test\_chromo[time\_max-1]=test\_chromo[time\_max-1]+1; //adds a one to the end of the chromosome
    for (i=time\_max-1;i>=1;i=i-1)
    {
        if (test\_chromo[i]>=2)
        {
            test\_chromo[i]=0;
            test\_chromo[i-1]=test\_chromo[i-1]+1;
            //getch();
            //gotoxy(10,19);
            //x\_print();
        }
    }
    if (test\_chromo[0]>1)
        return 1;
    else
        return 0;
}

/*/void x\_test(void)
{

}
float best, current, add;
int i;
x_initial();
best=x_eval();
gotoxy(10,18);
printf("X-Test Best: ");
while ((x_addition()) == 0) {
    current = x_eval();
    gotoxy(10,21);
    printf("Current is below: %f", current);
    gotoxy(10,22);
    x_print();
    if (best > current)
    {
        best = current;
        //getch();
        gotoxy(10,19);
        x_print();
    }
    //for(i=0;i<=time_max-1;i++){
    //    printf("%d",test_chromo[i]);
    //}
    gotoxy(26,18);
    printf("%8.1f", best);
}

}

void run_time(void)
{
    end = clock();
    printf("%.1f", (end - start) / CLK_TCK);
}

void rand_list(void)
{
    int i, loca_1, loca_2, temp_of_loca_1;

    for (i=0;i<=size-1;i++){
        list[i]=i;
    }

    for (i=1;i<=4*size;i++){
        loca_1 = u_rand(size);
        loca_2 = u_rand(size);
    }
void crossover(int a)
{
    int temp;
    float temp2;
    int pos,i;
    int average_tank, average_pump;

    int average_ratio;

    //randomize();
    pos=random(time_max-2);

    //the everaging of the ratios
    average_ratio=(second_chromo[list[a]][0]+second_chromo[list[a+1]][0])/2;
    second_chromo[list[a]][0]=average_ratio;
    second_chromo[list[a+1]][0]=average_ratio;

    average_tank=(second_chromo[list[a]][1]+second_chromo[list[a+1]][1])/2;
    second_chromo[list[a]][1]=average_tank;
    second_chromo[list[a+1]][1]=average_tank;

    average_pump=(second_chromo[list[a]][2]+second_chromo[list[a+1]][2])/2;
    second_chromo[list[a]][2]=average_pump;
    second_chromo[list[a+1]][2]=average_pump;

    for (i=pos;i<=time_max-1;i++)
    {
        temp=orig_chromo[list[a]][i];
        orig_chromo[list[a]][i]=orig_chromo[list[a+1]][i];
        orig_chromo[list[a+1]][i]=temp;

        /*temp2=storage_specs[list[a]][1];
        storage_specs[list[a]][1]=storage_specs[list[a+1]][1];
        storage_specs[list[a+1]][1]=temp2;*/
    }
}
void show_done(void)
{
    gotoxy(45,10);
    printf("done ");
}

void show_pair(void)
{
    gotoxy(45,10);
    printf("pairing ");
}

void show_mutation(void)
{
    gotoxy(45,10);
    printf("mutating ");
}

void show_evaluation(void)
{
    gotoxy(45,10);
    printf("evaluating ");
}

void show_crossover(void)
{
    gotoxy(45,10);
    printf("crossover ");
}

void show_generation(int a)
{
    gotoxy(25,10);
    printf("%d ",a);
}

void expand_price(void)
{
    int i,j,c;
    c=24;
    if (time_max>24)
        for (i=time_max-1;i>=0;i=i-(60/time_minutes))
        {
            c=c-1;
            for(j=0;j<(60/time_minutes);++j)
            {
                price[i-j]=price[c];
            }
        }
void expand_demand(void)
{
    int i, j, c;
    c = 24;

    if (time_max > 24) {
        for (i = time_max - 1; i >= 0; i = i - (60/time_minutes)) {
            //printf("n probability %f %d", prob, c);
            if (time_max > 24) {
                for (i = time_max - 1; i >= 0; i = i - (60/time_minutes)) {
                    //printf("n probability %f %d", prob, c);
                    c = c - 1;
                    for (j = 0; j < (60/time_minutes); ++j) {
                        demand[i - j] = demand[c];
                    }
                }
            }
        }
    }
}

/* float final_storage(int i) */
{
    int j;
    storage_remaining[0] = storage_specs[i][1] * storage_max;
    for (j = 1; j <= time_max; j++) {
        storage_remaining[j] = storage_remaining[j - 1] + (Q*orig_chromo[i][j - 1] - demand[j - 1]) * (3600/(60/time_minutes));
        if (storage_remaining[j] > storage_max) {
            storage_remaining[j] = storage_max;
        }
    }
    printf("%d %f", j, storage_remaining[j]);
}

return storage_remaining[time_max];
}

void x_print(void)
{
    int j;
    for (j = 0; j <= time_max - 1; j++) {
        printf("%d", test_chromo[j]);
    }
}
void eval(void)
{
    int i, j;
    float x;
    for (i=0; i<=size-1; i++){
        order_of_evaluation[i]=i;
        evaluate[i]=0;
        x=0;

        storage_remaining[0]=second_chromo[i][0]*tanks[second_chromo[i][1]][2]/1000;

        x=orig_chromo[i][0]*price[0]*(pumps[second_chromo[i][2]][3]*pumps[second_chromo[i][2]][2]
            *9.8/(pumps[second_chromo[i][2]][4])/(60/time_minutes);

        for (j=1; j<=time_max; j++){
            storage_remaining[i]=storage_remaining[j-1] +
                (pumps[second_chromo[i][2]][2]*orig_chromo[i][j-1] - demand[j-1])*60*time_minutes;

            if (j<time_max){
                x= x +
                    orig_chromo[i][j]*price[j]*(pumps[second_chromo[i][2]][3]*pumps[second_chromo[i][2]][2]
                        *9.8/(pumps[second_chromo[i][2]][4])/(60/time_minutes);
            }
        }

        if (storage_remaining[j] > tanks[second_chromo[i][1]][2]){
            storage_remaining[j] = tanks[second_chromo[i][1]][2];
        }

        if (storage_remaining[j] <= 0){
            x= x + 10*(0 - storage_remaining[j]);
        }

        if ((j==time_max) && (storage_remaining[j] <
                second_chromo[i][0]*tanks[second_chromo[i][1]][2]/1000)) {
            x = x +
                2*(second_chromo[i][0]*tanks[second_chromo[i][1]][2]/1000 - storage_remaining[j]);
        }
    }
}

125
evaluate[i]=tanks[second_chromo[i][1]][0] +
pumps[second_chromo[i][2]][0] + 
P_{given_A} *(365*x + tanks[second_chromo[i][1]][0]*tanks[second_chromo[i][1]][1] +
pumps[second_chromo[i][2]][0]*pumps[second_chromo[i][2]][1]);

}/*Boat x-eval(void)
{
  int j;
  float x;
  for (j=0; j<=time_max; j++) {
    storage_remaining[j] = 0;
  }
  //gotoxy(10,22);
  //x_print();
  //gotoxy(10,27);
  x = 0;
  storage_remaining[0] = storage_specs[0][1]*storage_max; // needs changing
  x = test_chromo[0]*price[0]*(H*Q*9.8/n)/(60/time_minutes);

  for (j=1; j<=time_max; j++) {
    storage_remaining[j] = storage_remaining[j-1] + (Q*test_chromo[j-1] - demand[j-1])*60*time_minutes;
    //printf("%d\n", j);
    if (j<time_max) {
      x = x + test_chromo[j]*price[j]*(H*Q*9.8/n)/(60/time_minutes);
    }
    if (storage_remaining[j] > storage_max) {
      storage_remaining[j] = storage_max;
    }
    if (storage_remaining[j]<=0) {
      x = x + 2*(0 - storage_remaining[j]);
    }
    if ((j==time_max) && (storage_remaining[j] < storage_specs[0][1]*storage_max)) {

void print_list(void)
{
    int i;

    for (i=1;i<=size;i++)
    {
        printf("\n%d %d",i,iistb[j]);
    }
}

float unit(void)
{
    float i;
    float x;
    i=rand();
    x=i/RAND_MAX;
    return x;
}

void mutation(float x)
{
    int i,j,number_chromo,a,b,c,d,e,f,g,temp,t,number_storage_mut;
    number_chromo=(size*x)/1;
    number_storage_mut=(size*storage_mut)/1;
    //printf("\n in size %d",size);
    //printf("\n in probability %f",x);
    //printf("\n in probability %f",prob);
    //printf("\n in mutation %d",number_chromo);
}

x = x + 2*(storage_specs[0][1]*storage_max - storage_remaining[j]);

//gotoxy(10,25);
//printf("%f",x);
for (i=0;i<=number_chromo;i++){
    //gotoxy(0,50);
    for (j=1;j<=changes;j++){
        a=random(size);
        b=random(time_max); // (time_max);
        c=random(time_max); // (time_max),
        /*printf("\nbefore number %d %d %d %d
",a,b,c,orig_chromo[a][b],orig_chromo[a][c]);
        for (t=0;t<=time_max-1;t++){
            printf("%d",orig_chromo[a][t]);
        }*/
        temp=orig_chromo[a][b];
        orig_chromo[a][b]=orig_chromo[a][c];
        orig_chromo[a][c]=temp;
        /*printf("\nafter number %d location1 %d location2 %d v1 %d v2 %d \n",a,b,c,orig_chromo[a][b],orig_chromo[a][c]);
        for (t=0;t<=time_max-1;t++){
            printf("%d",orig_chromo[a][t]);
        }*/
        /* */
    }
}
for (i=0;i<=number_chromo;i++){
    a=random(size);
    b=random(time_max); // (time_max);
    if (orig_chromo[a][b]==1)
        orig_chromo[a][b]=0;
    else
        orig_chromo[a][b]=0;
}
for (i=0;i<=number_storage_mut;i++){
    c=random(size);
    second_chromo[c][0]=random(1001);
}
for (i=0;i<=number_storage_mut;i++){
    d=random(size);
    // e=random(num_of_tanks+1);
    second_chromo[d][1]=random(num_of_tanks);
void roulette_wheel(void)
{
    int i,j,k,counter_inside_while;
    float random_number;
    gotoxy(10,21);
    //eval();

    running_total[0]=0;
    random_number=0;

    for (i=0;i<=size-1;i++){
        running_total[i]=0; //initiate
    }

    running_total[0]=1000000/pow(evaluate[0],Po);

    for (i=1;i<=size-1;i++){
        running_total[i]=running_total[i-1] + 1000000/pow(evaluate[i],Po);
    }

    for (i=0;i<=size-1;i++){
        counter_inside_while=size-1;
        random_number = unit()*running_total[size-1];

        for(k=1;k<=size-2;k++){
            if ((random_number > running_total[k])&& (random_number < running_total[k+1])){
                counter_inside_while=k+1;
            }
        }
    }

    if (random_number<running_total[0]){

counter_inside_while=0;
}

for (j=0;j<=time_max-1;j++){
    new_chromo_list[i][j]=orig_chromo[counter_inside_while][j];
}
for (j=0;j<=2;j++){
    second_chromo_list[i][j]=second_chromo[counter_inside_while][j];
}

//new_storage_specs[i][1]=[counter_inside_while][1];
}
for (i=0;i<=size-1;i++){
    for (j=0;j<=time_max-1;j++){
        orig_chromo[i][j]=new_chromo_list[i][j];
    }
    for (j=0;j<=2;j++){
        second_chromo[i][j]=second_chromo_list[i][j];
    }
    //storage_specs[i][1]=new_storage_specs[i][1];
}

void qsort(float v[],int left, int right)
{
    int i, last,last2;

    if (left >=right)
        return ;
    swap(v,left,(left + right)/2);
    swap2(order_of_evaluation,left,(left + right)/2);
    last=left;
    last2=left;
    for(i=left+1;i<=right;i++){
        if (v[i]< v[left]){
            swap(v, ++last,i);
            swap2(order_of_evaluation, ++last2,i);
        }
    }
```c

void swap(float v[], int i, int j)
{
    float temp;
    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}

void qsort2(float v[], int left, int right)
{
    int i, last,last2;

    if (left >= right)
        return;
    swap(v, left, (left + right)/2);
    swap2(order_of_evaluation, left, (left + right)/2);
    last=left;
    last2=left;
    for(i=left+1;i<=right;i++){
        if (v[i]< v[left]){
            swap(v, ++last, i);
            swap2(order_of_evaluation, ++last2, i);
        }
    }
    swap(v, left, last);
    swap2(order_of_evaluation, left, last2);
    qsort(v, left, last-1);
    qsort(v, last+1, right);
}

```

```
    v[j] = temp;

}

void children(void)
{
    int i,j;

    //this is new
    for (i=0; i<=9; i++) {
        for (j=0; j<=time_max-1; j++) {

            orig_chromo[order_of_evaluation[i+90]][j]=orig_chromo[order_of_evaluation[0]][j];

            //storage_specs[order_of_evaluation[i+90]][1]=storage_specs[order_of_evaluation[0]][1];

        }
    }

    //new ends here

    //everything below is the way it used to be
    /*for (i=1; i<=9; i++) {
        for (j=0; j<=time_max-1; j++) {

            orig_chromo[order_of_evaluation[i+49]][j]=orig_chromo[order_of_evaluation[i]][j];

            orig_chromo[order_of_evaluation[i+58]][j]=orig_chromo[order_of_evaluation[i]][j];

            orig_chromo[order_of_evaluation[i+67]][j]=orig_chromo[order_of_evaluation[i]][j];

        }
    }
    for (i=0; i<=9; i++) {
        for (j=0; j<=time_max-1; j++) {

            orig_chromo[order_of_evaluation[i+77]][j]=orig_chromo[order_of_evaluation[i+10]][j];

        }
    }
    for (i=0; i<=12; i++) {
        for (j=0; j<=time_max-1; j++) {

orig_chromo[order_of_evaluation[i+87]][j]=orig_chromo[order_of_evaluation[0]]
Appendix 10 - The Roulette Wheel Subroutine

The purpose of this appendix is to provide the reader with a comprehensive review of the development of the roulette wheel subroutine as used in GAHOS-1 and GAHOS-2. This appendix is divided into three parts. 1) A discussion of how the Roulette Wheel Method (RWM) is used in systems that require a maximization of their measured characteristics (the definition of measured characteristics will follow in the next paragraph). An example of a maximization problem would one where the measured characteristic is life expectancy or net profit. 2) A discussion of the RWM when used in systems that require a minimization of their measured characteristic (like total project cost as defined in GAHOS-2). 3) A review of magnification methods (within the roulette wheel subroutine) used in GAHOS-1 and GAHOS-2.

In section 3.6, the roulette wheel method of selecting children chromosomes for the roll of parent chromosomes was introduced. The main rival of the roulette wheel method is the chromosome ranking method. The ranking method’s decision structure is based upon the relevant ranking of a particular chromosome with respect to the rest of the population. The process is quite simple, all the chromosomes in a population are evaluated and ranked according to their fitness value. Afterwards duplicates of the top ranking chromosomes are made (i.e., stored in the computer’s memory) and in the process erase the lower ranking chromosomes. The number of duplicates made is specified by the user. For example, in a population of 100 chromosomes, one possible duplication mandate may be:

30 duplicates of 1st rank
20 duplicates of 2\textsuperscript{nd} rank
15 duplicates of 3\textsuperscript{rd} rank
12 duplicates of 4\textsuperscript{th} rank
10 duplicates of 5\textsuperscript{th} rank
7 duplicates of 6\textsuperscript{th} rank

This is a total of 94 duplicates plus the original 6 chromosomes produces a total of 100 chromosomes. The ranking method neglects (by definition) to take into account the degree of difference between the values being ranked. The only thing that is important is that there is a difference.

In the above mentioned example, all chromosomes ranked 7\textsuperscript{th} or lower were discarded and thus were never able to pass on their genes. This would be true even if the difference between the 1\textsuperscript{st} ranked chromosome and the 7\textsuperscript{th} ranked chromosome was only a fraction of a percent. There is thus the possibility that potentially useful chromosomes are discarded, simply because of their ranking and not as a result of their actual fitness value.

It is the above mentioned drawback that made the ranking method undesirable to use in GAHOS. The roulette wheel method was chosen for the roll of parent selection since this method is completely based on the fitness values relative to the rest of the population. This property of the roulette wheel method makes it an ideal candidate for the parent selection purposes.

A10.1 Maximizing Measured Characteristic
The term “fitness” is used to describe the ability of a chromosome to cope with its environment. The higher the fitness value, the superior the chromosome (i.e., solution) and vice versa. In nature the fitness of a chromosome, whether it may be a lion or a worm is the product of multiple characteristics. One can reasonability speculate that a lion’s fitness is composed of its

- ability to hunt,
- ability to defend its pride from other lions,
- ability to fend off other predators (i.e., hyenas),
- ability to reproduce, and its
- ability to wake up when laying in the path of an elephant.

One can see how difficult it is to assign a fitness value to a chromosome when the number of interactions between the chromosome and its environment are so vast.

Typically Genetic Algorithms avoid this complexity by defining the fitness as being related to only one measured characteristic. In GAHOS-1 this measured characteristic is daily electricity cost, while the measured characteristic in GAHOS-2 is total project cost. Other systems may define the measured characteristic as cost, reliability, efficiency, profit, capacity, etc. When the characteristic is directionally related to the fitness, than the problem is one of maximizing this characteristic. The phrase “directionally related” is used to describe a characteristic that is related in direction to the fitness (both the fitness and characteristic belong to the same chromosome). A simple way to identify a directionally related characteristic is to ask whether the fitness should increase as the values of the characteristic increases. Properties such as reliability,
efficiency, capacity, survivability, life expectancy are directionally related since they move in the same direction as their fitness value.

As mentioned above, an increase in a directionally related characteristics would cause the fitness value to increase also. Some examples of this are shown below, with the measured characteristic being storage tank capacity:

- Fitness = Capacity  (directionally related and directly proportional)
- Fitness = 0.75•Capacity  (directionally related and directly proportional)
- Fitness = (Capacity)²  (directionally related but not directly proportional)

Whether to make the characteristic directly proportional or not is left to the discretion of the application user, although some form of sensitivity analysis may aid this investigation.

Let's investigate how the roulette wheel method functions for a problem where the measured characteristic is both directionally related and directly proportional to the fitness value (it might be useful to the reader to review section 3.6.1 first).

Fitness = Characteristic  (directionally related and directly proportional)

We assume there are \( n \) chromosomes in the population, and assemble the following table to be used for the roulette wheel method. The roulette wheel method requires the creation of a uniformly distributed random number between zero and \( \sum_{i=1}^{n} F_i \). Therefore using the same technique that was used in Fig. 3.10, one can create the table shown on the top of the next page.
<table>
<thead>
<tr>
<th>Chromosome</th>
<th>1, 2 ........ j ........ n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristic</td>
<td>C₁, C₂ ........ C_j ........ C_n</td>
</tr>
<tr>
<td>Fitness</td>
<td>F₁, F₂ ........ F_j ........ F_n</td>
</tr>
</tbody>
</table>

Running Total | F₁, F₁ + F₂ ........ \( \sum_{i=1}^{j} F_i \) ........ \( \sum_{i=1}^{n} F_i \)

Thus the probability of the \( k^{th} \) chromosome being chosen is:

\[
\text{Prob.} \ (k) = \frac{\sum_{i=1}^{k} F_i - \sum_{i=1}^{k-1} F_i}{\sum_{i=1}^{n} F_i}, \quad k < n
\]

since \( \sum_{i=1}^{k} F_i = \sum_{i=1}^{k-1} F_i + F_k \)

\[
\therefore \text{Prob.} \ (k) = \frac{\sum_{i=1}^{k-1} F_i + F_k - \sum_{i=1}^{k-1} F_i}{\sum_{i=1}^{n} F_i}
\]

\[
\text{Prob.} \ (k) = \frac{F_k}{\sum_{i=1}^{n} F_i}, \quad k < n
\]

Since \( \sum_{i=1}^{n} F_i \) is a constant for any given generation. It can therefore be said that the probability of a \( k^{th} \) chromosome being chosen in any generation is directly proportional to its fitness. In other words:

If Fitness \( (k) = \text{Characteristic} \ (k) \)

then Prob. \( (k) \propto \text{Characteristic} \ (k) \)

Using the same technique used to derive the above relationship, a few other examples these relationships are provided below:

If Fitness \( (k) = 0.75 \times \text{Characteristic} \ (k) \)

then Prob. \( (k) \propto \text{Characteristic} \ (k) \)
If Fitness \((k) = [\text{Characteristic } (k)]^2\)

then Prob. \((k) \propto [\text{Characteristic } (k)]^2\)

If Fitness \((k) = 0.75[\text{Characteristic } (k)]^2 + e^{2.35}\)

then Prob. \((k) \propto [\text{Characteristic } (k)]^2\)

The examples provided above were not selected for any specific reason, simply because they encompass the more common types of fitness-characteristic relationships (i.e., exponents, addition of constants, multiplication)

**A10.2 Minimizing Measured Characteristic**

Appendix 10.1 discussed scenarios in which the fitness value was directionally related to the property (i.e., characteristic) measured. These types of scenarios are maximization focused, since they try to find chromosomes that maximize the desired characteristic.

When the property measured is counter-related to the fitness, than the roulette wheel method is used to minimize that property. The measured characteristics in GAHOS is daily electricity cost or total project cost which are both counter related characteristics.

The simplest relationship that can be drawn from the fitness value and cost is stated below:

\[
\text{Fitness } (k) = [\text{cost } (k)]^{-1} = \frac{1}{\text{cost}(k)}
\]

This type of formulation results in the following probability:
Beckwith and Wong (1995) as well as Wong and Wong (1994) recommend using a different relationship between fitness and cost when analyzing water distribution systems. They suggest multiplying the inverse of the cost value by a largest possible constant. Their formula is shown below as:

\[
\text{Fitness } (k) = M \cdot [\text{cost } (k)]^{-1} = \frac{M}{\text{cost}(k)}
\]

where \( M \) = the maximum floating point number that can be represented by the computer

The constant \( M \) does not alter any of the probabilities, and is used for the sole purpose of magnifying the fitness values of the entire population. The proof of this is shown in the following equations:

If Fitness \( (k) = \frac{M}{\text{cost}(k)} \)

\[ \therefore \text{Prob. } (k) = \frac{M}{\sum_{i=1}^{n} \frac{M}{\text{cost}(i)}} \]

\[ = \frac{1}{\sum_{i=1}^{n} \frac{1}{\text{cost}(k)}} \cdot \frac{1}{\sum_{i=1}^{n} \frac{1}{\text{cost}(i)}} \cdot M \cdot 1 \]

\[ = \frac{1}{\sum_{i=1}^{n} \frac{1}{\text{cost}(i)} \cdot \text{cost}(k)} \]
As can be seen, M does not effect the probability distribution since it is present in both the nominator and denominator and therefore cancels out.

A magnification factor such as M can be useful. However, declaring M to be the maximum floating point number (Beckwith and Wong (1995) as well as Wong and Wong (1994) did) is dangerous with respect to the operation of the computer. To see this, consider that at low cost(κ) values and high population sizes (i.e., large n), the possibility exists that the running total may exceed the value of M. Since M is the largest possible number the computer can manipulate, there exits a possibility of an overflow or erroneous calculation (the computer could store larger numbers if they were defined as double floating points). This malfunction can take one of two forms. The first would be that the computer simply stalls without warning and with no suggestions of cause other than the possibility of an error message describing a floating point error. Second, the computer may simply stop counting at M and just assign the value of M to any function that attempts to surpass it. This would be troublesome since it would not be readily evident that a malfunction has taken place. Indirect evidence may be found from the distribution of chromosomes chosen by the roulette wheel method. The method would be biased towards the lower chromosome numbers, specifically the chromosomes which have numbers between 1 and T. Where T is defined as:

\[
T = \frac{1}{\sum_{i=1}^{\infty} \frac{M}{\text{cost}(i)}} \leq M, \quad T \in \mathbb{R}, T \lt n
\]
The above discussion has focused on the dangers of valuing $M$ as the largest floating point integer the computer allows. However, these same dangers can also occur when $M$ is less than the maximum floating point integer. Low cost values and large population sizes can also lead to the same types of computer errors discussed previously. It is thus up to the user to ensure that the magnification factor (i.e., $M$) is customized with respect to the population size and the system (the system which is being optimized).

A10.3 Magnification Methods Used in GAHOS

Questions arises about when a magnification factor (i.e., $M$) should be used and what value of $M$ should be assigned. Theoretically the magnification factor should be used when the original range of fitness values lies outside the possible storage or functional domain of the computer. For example, the Borland C++ compiler allows the storage of floating point numbers in the range of $3.4 \times 10^{-38}$ to $3.4 \times 10^{38}$ (Kernighan and Ritchie, 1988). The function domain is the range in which the computer can mathematically manipulate the number. For example, if the original fitness values ranged from $5 \times 10^{-20}$ to $3 \times 10^{-40}$, the lower part of the fitness value distribution will be adversely affected by the computer's capability. Thus we can conclude that a magnification factor (i.e., $M$) should be used. The value of $M$ should be such that it draws the range of fitness values into the computer's floating point domain.

Both versions of GAHOS when utilizing input parameters (i.e., water demand, electricity cost, $P$ value, population size, pump cost, storage tank cost) which are similar to those available in the appendices do not require a magnification factor. However, if the input parameters or the system as defined were to change drastically, a magnification
factor might be necessary. Under such circumstances the equation presented below may be used to choose an adequate M value:

\[
\frac{3.4}{\text{Average(fitness)}},
\]

This equation will determine the best M value in a computer with a functional domain range of \(3.4 \times 10^{-38}\) to \(3.4 \times 10^{38}\). The user should note that the center value of the functional domain range is not \(1.7 \times 10^{-38}\) (The average between the two extremes), but \(3.4 \times 10^0\) or simply 3.4.

A second magnification factor was developed for use in GAHOS, it is called the magnification exponent P (not the same as P the mutation probability per chromosome as used in the ICMM). The magnification exponent is used in the fitness value expressions as shown below:

\[
\text{Fitness} (k) = \frac{M}{\text{cost(k)}^P}
\]

\[
\text{Prob.} (k) = \frac{1}{\sum_{i=1}^{n} \left( \frac{1}{\text{cost(i)}^P} \right)} \cdot \text{cost(k)}^P
\]

When P is less than unity, the roulette wheel method reduces the ability of low cost chromosomes to outperform high cost chromosomes. In other words the sensitivity of the method is decreased (the roulette wheel is less biased than when P=1). When P is greater than unity than the ability of low cost chromosomes to outperform high cost chromosomes is increased. In other words the roulette wheel becomes more and more biased towards low cost chromosomes as the value of P increases.
Fig. A10.1 presented below demonstrates how varying the magnification exponent alters the probability distribution of population of chromosomes. For example chromosome No. 3 has a greater probability of passing on its genes as the value of P decreases. This is because chromosome No. 3 is a low performing chromosome, and smaller P values decrease the biased property of the roulette wheel.

**Table A10.1: The effect of the Magnification exponent**

<table>
<thead>
<tr>
<th>Chromosome No.</th>
<th>Cost [$]</th>
<th>Probability when P = 0.5</th>
<th>Probability when P = 1</th>
<th>Probability when P = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.4377</td>
<td>0.5454</td>
<td>0.7347</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0.3095</td>
<td>0.2727</td>
<td>0.1837</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>0.2527</td>
<td>0.1818</td>
<td>0.08163</td>
</tr>
</tbody>
</table>

Chromosome No. 1 is the best performing individual in the entire population. Thus it will always have the highest probability of passing on its genes. However, the lower the P value, the less of an advantage it has over the rest of the chromosome in its population.

Natural evolution and GA both share the property of replicating high performance chromosomes in subsequent generations. The magnification exponent simply allows the GA user to increase or decrease the effectiveness of that property.

It is important to remember that the mutation probability (P), magnification factor (M), magnification exponent (P) value and other GA parameters are set at levels that decrease the length of the simulation and increase its confidence limits (increase the certainty that the program found the best solution once the simulation has ended).
One way of evaluating the effectiveness of the genetic algorithms parameters is to look at its $G_P$ value. The $G_P$ value is equal to the number of generations the program had to run before it found the absolute optimal solution. By running the program over a long period of time (between 5000 and 15000 generation) one can find the optimal solution. The next step is to alter the GA parameters so as to decrease the $G_P$ value. Fig. A10.1 shown below illustrates the relationship between the $G_P$ value and cost (which in this case is actually daily electricity cost).

**Fig. A10.1:** Daily electricity cost vs. Number of generations

![Graph: Daily electricity cost vs. Number of generations](image)

The $G_P$ value of the above figure is equal to 173. In other words this particular simulation reached the optimal solution (i.e., $53.71/day) at the 173^{rd} generation.

One point the reader should note is that a good set of GA parameters is found by striving for a low $G_P$ average (take the average $G_P$ value of at least 30-50 runs) and low standard deviation. Finding a good set of GA parameters this way requires a sizable investment of time (30 to 50 runs per set of parameters). However, once this set is found
it is suitable for all simulations that utilize similar input (similar: population size, pumps, storage tanks, electricity rate structure, water demand profile).
Appendix 11 - Starting Storage to Capacity Ratio

A11.1 Starting Storage Values

The amount of water present in the water reservoir at the start of a simulation can have a profound effect on the daily electricity cost (i.e., the solution recommended by GAHOS changes depending on the starting volume). The starting storage to maximum capacity is referred to as the SCR (Starting to Capacity Ratio). The SCR was found to be a critical parameter in both GAHOS-1 and GAHOS-2.

The importance of the SCR in the course of finding the optimal solution was a complete surprise. Originally GAHOS-1 was designed to run using a constant SCR. This constant would be used for all storage tank sizes and for all pumps (an SCR of between 0.4 to 0.6 was contemplated). However, it was soon evident that this simplification (using a constant SCR for all simulations) would not be possible. As will be described in the coming pages the SCR was made a variable that is optimized by GAHOS, much like the way the pumping schedule to optimized.

It is first important to study how the SCR effects the operation of the water distribution system. Every combination of storage tank size and pumping system (as well as water demand profile and electricity cost rate structure) has an optimal SCR. There are two principles that govern what the optimal SCR is.

A) The ability of the reservoir to supply water to the network during the peak energy charge hours (i.e., minimizing peak-period pumping).
B) The ability of the reservoir to take advantage of the off-peak period (i.e., maximizing off-peak-period pumping).

An effective way of demonstrating these two principles is to imagine a water distribution system with a fixed storage size and pumping station. What is the optimal SCR for this system? The answer depends on where time zero is defined in energy charge cycle. The energy charge cycle is defined as the cycle consisting of the peak period and off-peak period. It is quite reasonable to assume that these two periods are 12 hours in length (i.e., a total of 24 hours). If time zero is located near the end of the off-peak period or the beginning of the peak-period, then the SCR would be high. This is because principle A is the prevailing influence. However, if time zero is located near the end of the peak-period or the beginning of the off-peak period then principle B would prevail and the S/C ratio would be low.

As can be seen from the energy charge cycle depicted in Fig. A11.1, the optimal SCR is strongly dependent on where time zero is defined. However, the optimal SCR is not exclusively dependent on the current position in the energy charge cycle. Other factors such as water demand profile, storage capacity and pump characteristics can also affect the optimal SCR (Chase and Ormsbee, 1993). In other words, for every combination of demand profile, storage capacity, pumping system and schedule there exists a SCR that minimizes the operational costs. Since the computer program is designed to optimize the water distribution system, it makes sense to allow the program to optimize the SCR also.
As stated previously the original design of GAHOS-1 specified that the SCR was not an optimizable variable but rather a user specified property (like the water demand profile or electricity prices). By running a preliminary version of GAHOS-1 (one in which the SCR is user defined) many times using a broad range of SCRs, Fig. A11.2 was created (Appendix 1 describes the experiment in detail). The experiment produced some interesting results. As can be seen from Fig. A11.2, there exists a range of SCRs that produce the lowest daily pumping costs. The reader should keep in mind that every data point in Fig. A11.2 is a combination of an SCR value and pumping schedule with an associated daily electricity cost. The pumping schedule does change as the SCR varies. This even occurs in the optimal range of SCRs (i.e., between 0.005 and 0.68).

The existence of a valley or range of optimal SCRs makes intrinsic sense, especially when analyzed using the two governing principals mentioned previously. At very low
Fig. A11.2: Daily Pumping Costs vs. S/C ratio

Optimal S/C ratio is between 0.5% and 68%

SCRs, the reservoir may not be able to supply water to the network during the peak energy charge hours without the use of the pumps. At very high SCRs, the reservoir capacity remaining may not be sufficient to take advantage of the off-peak period.

Fig. A11.2 illustrates the complex nature of the SCR and how difficult it would be for the user to estimate what the SCR value should be. Similar experiments (similar to that belonging to Fig. A11.2) were conducted using several different storage tank sizes.

Table A11.1: Optimal range of S/C ratios for different reservoir sizes

<table>
<thead>
<tr>
<th>Maximum Storage Capacity [m³]</th>
<th>Daily Pumping Cost [S/day]</th>
<th>Optimal SCR range [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>133.4</td>
<td>90 - 95</td>
</tr>
<tr>
<td>2000</td>
<td>77.5</td>
<td>0.5 - 30</td>
</tr>
<tr>
<td>4000</td>
<td>57.1</td>
<td>5 - 10</td>
</tr>
<tr>
<td>8000</td>
<td>53.7</td>
<td>10 - 50</td>
</tr>
<tr>
<td>12000</td>
<td>53.7</td>
<td>0.5 - 68</td>
</tr>
<tr>
<td>∞</td>
<td>53.7</td>
<td>1/∞ to 100</td>
</tr>
</tbody>
</table>
By looking at the optimal range of SCRs for the various tank sizes it is evident how wrong it was to initially assume that the range would be between 0.4 and 0.6 (as previously mentioned).

The relationship between the optimal SCR range, daily pumping cost and storage size is not readily evident at first glance. There appears to be no correlation between the three, however relationship does exists. When the storage tank sizes are greater than 8,000 m$^3$ the daily pumping cost is the smallest. All tanks greater than 8,000 m$^3$ produce the same minimum daily pumping costs of $53.7/day. In fact, an infinitely large storage tank could not better that pumping cost. It is for this reason that the term “needlessly huge” can be used to describe all storage tanks greater than 8,000m$^3$. “Needlessly huge” because a smaller tank can equal the pumping cost and at the same time require less initial capital for its construction.

Let us imagine the smallest storage tank that can maintain a $53.7/day electricity charge. We will refer to the size of this storage tank as $X$. By viewing Table A11.1, we can conclude that $X$ is between 4,000 m$^3$ and 8,000 m$^3$. All storage tanks greater that $X$ have a broader SCR range than $X$ itself. The bigger the tank, the broader the optimal SCR range becomes. This can be seen by comparing the optimal SCR range for storage size of the 8,000m$^3$ and the 12,000m$^3$ storage tank (see Table A11.1). The 8,000m$^3$ tank has an optimal range between 0.1 and 0.5, while the 12,000m$^3$ tank has an optimal range between 0.005 and 0.68. In fact an infinitely large storage tank has a SCR range between $1/\infty$ and 1.
For tank sizes less than $X$, the relationship between the SCR and storage tank size becomes very complicated. Tank sizes below $X$ must find an equilibrium point that satisfy both Principal A and Principal B. The term “equilibrium point” is used to describe the set of conditions (i.e., SCR, pumping schedule etc.) that achieve the lowest possible cost. The equilibrium point between Principal A and Principal B is very sensitive to demand rates, electricity cost rates and differing pumping systems. Thus it is futile to explicitly state an SCR even if the storage tank capacity is fixed. The only logical conclusion is to make the SCR a decision variable that is optimized by the GA. Thus GAHOS-1 was designed to encode the SCR and pumping schedule in its chromosomes and optimize them simultaneously. While GAHOS-2 was designed to encode the SCR, pumping schedule, storage tank size and pump type in its chromosomes and solve these variables simultaneously also.
Appendix 12 - ENUMERO computer code

#include <conio.h>
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <time.h>
#include <dos.h>
#include <math.h>
#include <stdlib.h>

char ifile[10]; /*global variables*/
int size; /*global variables*/
int ngen; /*global variables*/
int time_minutes;
int time_max;
float prob;
int changes;
float demand[96]; /*global variables*/
float price[96]; /*global variables*/
int orig_chromo[100][96];
int test_chromo[96];
float evaluate[100];
float storage_remaining[97];
int list[100];
int new_chromo_list[100][96];
int order_of_evaluation[100];

void opening_file(void); /*function prototype*/
void output_file(void);
void screen_data(void); /*function prototype*/
void title(void); /*function prototype*/
void run_time(void);
void rand_list(void);
void print_list(void);
void crossover(int a);
void show_crossover_title(void);
void show_crossover(void);
void show_pair(void);
void show_mutation(void);
void show_evaluation(void);
void show_done(void);
void show_generation(int a);
void roulette_wheel(void);
void mutation(float x);
void qsort(float v[], int left, int right);
void qsort2(float v[], int left, int right);
void swap(float v[], int i, int j);
void swap2(int v[], int i, int j);
void expand_price(void);
void expand_demand(void);
float final_storage(int i);
int u_rand(int a);
float unit(void);
int random(int num);
void eval(void);
void children(void);
int x_addition(void);
float x_eval(void);
void x_test(void);
void x_initial(void);
void x_print(void);
float storage_max;
float storage_start;
float Q;
float H;
float n;
int best = 99;
float high = -999;

clock_t start, end;

void main(void) /* main program*/
{
    int i, j, p;
    randomize();
    start = clock();
    title();
    screen_data();
    title();
    opening_file();
    //printf("\n probability \%f", prob);
    //  getch();
    expand_price();
    //printf("\n probability \%f", prob);
    //  getch();
    expand_demand();
    //printf("\n probability \%f", prob);
    //  getch();
}
/*for (p=0;p<=ngen;p++){
    show_generation(p);
    rand_list();
    show_pair();
    if (p<ngen){
        for (i=0;i<=size-2;i=i+2){
            crossover(i);
            show_crossover();
        }
        show_mutation();
        mutation(prob);
        show_evaluation();
    }
    eval();
    qsort(evaluate,0,99);
}
if (p<ngen){
    children();
    mutation(prob);
    eval();
    qsort(evaluate,0,99);
}
gotoxy(10,14);
printf("Current best: %.1f ",evaluate[0]);
if (p==ngen){
    show_done();
    gotoxy(10,12);
    printf("time elapsed(sec): ");
    run_time();
    /*for (i=0;i<=time_max-1;i++){
        printf("\n%d %f %f",i,price[i],demand[i]);
    }*/
    //getch();
    /*for (i=0;i<=time_max-1;i++){
        printf("\n%.4f",orig_chromo[order_of_evaluation[j]][i]*price[i]*140.27);
    }*/
    //getch();
    gotoxy(10,16);
    //printf("Reservoir stage\n");
    for (j=0;j<=0;j++){
//printf("n%d %.1f %n",j,evaluate[j],final_storage(order_of_evaluation[j]));

printf("The best solution:");
gotoxy(10,17);
for (i=0;i<=time_max-1;i++){
    printf("%%d",orig_chromo[order_of_evaluation[0]][i]);
}
}
} */

//printf("n%d %d",random(time_max),random(time_max));
//output_file();

x_initial();
//x_addition();
x_test();
//x_eval();

/*while
gotoxy(10,24);
for (i=0;i<=23;i++){
    printf("%d",test_chromo[i]);
}
gotoxy(10,25);
i=x_eval();*/

void title(void)
{
    clrscr();
gotoxy(15,1);
    printf("ONE PUMP ONE RESERVOIR: VERSION 1.0");
}

void output_file(void)
{
    int i,j;
    FILE *stream2;
    stream2 = fopen("out.txt","w");
    for (i=0;i<=time_max-1;i++){
        fprintf(stream2,"\n%d %f %f",i,price[i],demand[i]);
    }
    fprintf(stream2,"\nThe best solution\n\n");
for (i=0; i<=time_max-1; i++){
    fprintf(stream2, "%d\t", orig_chromo[order_of_evaluation[0]][i]);
}
}

void opening_file(void)
{
    int i,j;
    FILE *stream1;
    stream1 = fopen(ifile, "r");
    gotoxy(10,5);
    printf("Opening file: %s", ifile);
    gotoxy(10,6); /*start location of line below*/
    printf("Reading file: ");
    for (i=0; i<=23; i++){
        fscanf(stream1, "%f\n", &price[i]);
        gotoxy(24,6); /*pretend info*/
        printf("%d", demand[i]);
    }
    for (i=0; i<=23; i++){
        fscanf(stream1, "%f\n", &demand[i]);
        gotoxy(24,6);
        printf("%f", price[i]);
    }
    gotoxy(10,6);
    printf("chromosomes: ");
    for (j=0; j<=size-1; j++){
        fscanf(stream1, "\n");
        gotoxy(23,6);
        for (i=0; i<=time_max-1; i++){
            fscanf(stream1, "%d", &orig_chromo[j][i]);
            if (time_minutes>15){
                printf("%d", orig_chromo[j][i]);
            }
        }
    }
    gotoxy(10,6);
    printf("chromosomes initiation: done ");
    gotoxy(10,10);
    printf("generation #: ");
    gotoxy(35,10);
    printf("status: ");
}
void screen_data(void)
{

gotoxy(1,4);
printf("Enter input file name: ");
gets(ifile);
gotoxy(1,5);
printf("Enter the time steps (minutes): ");
scanf("%d",&time_minutes);
gotoxy(1,6);
printf("Enter the population size: ");
scanf("%d",&size);
gotoxy(1,7);
printf("Enter the number of generations: ");
scanf("%d",&ngen);
gotoxy(1,8);
printf("Enter the probability of mutation: ");
scanf("%f",&prob);
gotoxy(1,9);
printf("Enter the mutation per chromosome: ");
scanf("%d",&changes);
gotoxy(1,10);
printf("Enter the storage capacity: ");
scanf("%f",&storage_max);
gotoxy(1,11);
printf("Enter the starting storage: ");
scanf("%f",&storage_start);
gotoxy(1,12);
printf("Enter the pump flow: ");
scanf("%f",&Q);
gotoxy(1,13);
printf("Enter the pump head: ");
scanf("%f",&H);
gotoxy(1,14);
printf("Enter the pump efficiency: ");
scanf("%f",&n);
time_max=(60/time_minutes)*24;
}

void x_initial(void)
{
    int i;
    for (i=0;i<=time_max-1;i++)
    {
        test_chromo[i]=0;
    
}
```c
int x_addition(void)
{
    int i;
    test_chromo[time_max-1]=test_chromo[time_max-1]+1; // adds a one to the end of the chromosome
    for (i=time_max-1;i>=1;i=i-1)
    {
        if (test_chromo[i]>=2)
        {
            test_chromo[i]=0;
            test_chromo[i-1]=test_chromo[i-1]+1;
            // getch();
            gotoxy(10,19);
            x_print();
        }
    }
    if (test_chromo[0]>1)
        return 1;
    else
        return 0;
}

void x_test(void)
{
    float best,current,add;
    int i;
    x_initial();
    best=x_eval();
    gotoxy(10,18);
    printf("X-Test: Best: ");
    while ((x_addition()==0)){
        current=x_eval();
        gotoxy(10,21);
        printf("current is below: %f",current);
        gotoxy(10,22);
        x_print();
        if (best>current)
        {
            best = current;
            // getch();
            gotoxy(10,16);
        }
    }
}
```
void run_time(void)
 {
     end = clock();
     printf("%.1f", (end - start) / CLK_TCK);
 }

void rand_list(void)
 {
     int i, loca_1, loca_2, temp_of_loca_1;

     for (i=0;i<=size-1;i++){
        list[i]=i;
     }

     for (i=1;i<=4*size;i++){
        loca_1=r_rand(size);
        loca_2=r_rand(size);
        temp_of_loca_1=list[loca_1];
        list[loca_1]=list[loca_2];
        list[loca_2]=temp_of_loca_1;
     }
 }

void crossover(int a)
 {
    int temp;
    int pos,i;
    //randomize();
    pos=random(22);

    for (i=pos;i<=time_max-1;i++){
        temp=orig_chromo[list[a]][i];
    }
}
void show_done(void)
{
    gotoxy(45,10);
    printf("done ");
}

void show_pair(void)
{
    gotoxy(45,10);
    printf("pairing ");
}

void show_mutation(void)
{
    gotoxy(45,10);
    printf("mutating ");
}

void show_evaluation(void)
{
    gotoxy(45,10);
    printf("evaluating ");
}

void show_crossover(void)
{
    gotoxy(45,10);
    printf("crossover ");
}

void show_generation(int a)
{
    gotoxy(25,10);
    printf("%d ",a);
}

void expand_price(void)
{
    int i,j,c;
    c=24;
    if (time_max>24){
        for (i=time_max-1;i>=0;i=i-(60/time_minutes)){
            orig_chromo[list[a]][i]=orig_chromo[list[a+1]][i];
            orig_chromo[list[a+1]][i]=temp;
        }
    }
}
```c
void expand_demand(void)
{
    int i,j,c;
    c=24;

    if (time_max>24)
    {
        for (i=time_max-1;i>=0;i=-(60/time_minutes))
            //printf("n probability %f %d",prob,c);
            //getchO;
        c=c-1;
        for(j=0;j<(60/time_minutes);++j) {
            demand[i-j]=demand[c];
        }
    }
}

float final_storage(int i)
{
    int j;
    storage_remaining[0]=storage_start;
    for (j=1;j<=time_max;j++){
        storage_remaining[j]=storage_remaining[j-1] + (Q*orig_chromo[i][j-1] - demand[j-1])*(3600/(60/time_minutes));
        if (storage_remaining[j] > storage_max){
            storage_remaining[j] = storage_max;
        }
        printf("n%d %f",j,storage_remaining[j]);
    }
    return storage_remaining[time_max];
}

void x_print(void)
{
    int j;
    for (j=0;j<=time_max-1;j++)
```
void eval(void)
{
    int i,j;
    float x;
    for (i=0;i<=size-1;i++)
    {
        order_of_evaluation[i]=i;
        evaluate[i]=0;
        x=0;
        storage_remaining[0]=storage_start;
        x=orig_chromo[i][0]*price[0]*(H*Q*9.8/n)/(60/time_minutes);

        for (j=1;j<=time_max;j++)
        {
            storage_remaining[j]=storage_remaining[j-1] +
            (Q*orig_chromo[i][j-1] - demand[j-1])*60*time_minutes;

            if (j<time_max)
            {
                x= x +
                orig_chromo[i][j]*price[j]*(H*Q*9.8/n)/(60/time_minutes);
            }

            if (storage_remaining[j] > storage_max)
            {
                storage_remaining[j] = storage_max;
            }

            if (storage_remaining[j]<=0)
            {
                x= x + 2*(0 - storage_remaining[j]);
            }

            if ((j==time_max) && (storage_remaining[j] < storage_start))
            {
                x = x + 2*(storage_start - storage_remaining[j]);
            }
        }

        evaluate[i]=x;
    }
}

float x_eval(void)
{  
    int j;
    float x;
    for (j=0;j<=time_max;j++){
        storage_remaining[j]=0;
    }
//gotoxy(10,22);
//x_print();
//gotoxy(10,27);
x=0;
    storage_remaining[0]=storage_start;
x=test_chromo[0]*price[0]*(H*Q*9.8/n)/(60/time_minutes);

    for (j=1;j<=time_max;j++){
        storage_remaining[j]=storage_remaining[j-1] + (Q*test_chromo[j-1] - demand[j-1])*60*time_minutes;
        //printf("%d\n",j);
        if (j<time_max){
            x = x +
test_chromo[j]*price[j]*(H*Q*9.8/n)/(60/time_minutes);
        }
        if (storage_remaining[j] > storage_max){
            storage_remaining[j] = storage_max;
        }
        if (storage_remaining[j]<=0){
            x = x + 2*(0 - storage_remaining[j]);
        }
        if ((j==time_max) && (storage_remaining[j] < storage_start)){
            x = x + 2*(storage_start - storage_remaining[j]);
        }
//gotoxy(10,25);
//printf("%f",x);
    }

    return x;
}

int u_rand(int a)
{
    return (random(a) + 1);
void print_list(void)
{
    int i;

    for (i=1; i<=size; i++) {
        printf("%d %d", i, list[i]);
    }
}

float unit(void)
{
    float i;
    float x;
    i = rand();
    x = i / RAND_MAX;
    return x;
}

void mutation(float x)
{
    int i, j, number_chromo, a, b, c, temp, t;
    number_chromo = (size * x) / 1;
    //printf("\n size %d", size);
    //printf("\n probability %f", x);
    //printf("\n probability %f", prob);
    //printf("\n mutation %d", number_chromo);
    for (i = 0; i <= number_chromo; i++) {
        //gotoxy(0, 50);

        for (j = 1; j <= changes; j++) {
            a = random(size);
            b = random(time_max); // (time_max);
            c = random(time_max); // (time_max);
            /* printf("\n before number %d %d %d %d %d %d a,b,c,orig_chromo[a][b],orig_chromo[a][c]);
            for (t = 0; t <= time_max - 1; t++) {
                printf("%d", orig_chromo[a][t]);
            } */
            temp = orig_chromo[a][b];
orig_chromo[a][b]=orig_chromo[a][c];
orig_chromo[a][c]=temp;

/*
 * 
 * 
 */

for (t=0;t<=time_max-1;t++){
    printf("%d",orig_chromo[a][t]);
}

for (i=0;i<=number_chromo;i++){
    a=random(size);
    b=random(time_max); //(time_max);
    if (orig_chromo[a][b]==1)
        orig_chromo[a][b]=0;
    else
        orig_chromo[a][b]=0;
}

void roulette_wheel(void)
{
    int i,j,counter_inside_while;
    int running_total[100];
    int random_number;

    running_total[0]=0;
    random_number=0;

    for (i=0;i<=size-1;i++){
        running_total[i]=0;
    }

    for (i=1;i<=size-1;i++){
        running_total[i]=running_total[i-1] + evaluate[i];
    }

    for (i=0;i<=size-1;i++){

        random_number = unit()*running_total[size-1];
        counter_inside_while=0;

        do {
            counter_inside_while=counter_inside_while+1;
        } while (running_total[counter_inside_while]<random_number);
for (j=0;j<time_minutes;j++){
    new_chromo_list[i][j]=orig_chromo[counter_inside_while][j];
}
for (i=0;i<size-1;i++){
    for (j=0;j<time_minutes;j++){
        orig_chromo[i][j]=new_chromo_list[i][j];
    }
}

void qsort(float v[],int left, int right)
{
    int i, last,last2;
    if (left >=right)
        return ;
    swap(v, left,(left + right)/2);
    swap2(order_of_evaluation, left,(left + right)/2);
    last=left;
    last2=left;
    for(i=left+1;i<right;i++){
        if (v[i]< v[left]){
            swap(v, ++last,i);
            swap2(order_of_evaluation, ++last2,i);
        }
    }
    swap(v, left, last);
    swap2(order_of_evaluation, left,last2);
    qsort(v, left, last-1);
    qsort(v, last+1, right);
}

void swap(float v[], int i, int j)
{
    float temp;
    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}

void qsort2(float v[],int left, int right)
{
    int i, last,last2;
}
if (left >= right)
    return;
swap(v, left, (left + right)/2);
swap2(order_of_evaluation, left, (left + right)/2);
last = left;
last2 = left;
for (i = left + 1; i <= right; i++) {
    if (v[i] < v[left]) {
        swap(v, ++last, i);
        swap2(order_of_evaluation, ++last2, i);
    }
}
swap(v, left, last);
swap2(order_of_evaluation, left, last2);
qsort(v, left, last-1);
qsort(v, last+1, right);

void swap2(int v[], int i, int j)
{
    int temp;
    temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}

void children(void)
{
    int i, j;
    for (i = 1; i <= 9; i++) {
        for (j = 0; j <= time_max-1; j++) {
            orig_chromo[order_of_evaluation[i+49]][j] = orig_chromo[order_of_evaluation[i]][j];
            orig_chromo[order_of_evaluation[i+58]][j] = orig_chromo[order_of_evaluation[i]][j];
            orig_chromo[order_of_evaluation[i+67]][j] = orig_chromo[order_of_evaluation[i]][j];
        }
    }
}
for (i=0; i<=9; i++)
    for (j=0; j<=time_max-1; j++)
        orig_chromo[order_of_evaluation[i+77]][j]=orig_chromo[order_of_evaluation[i+10]][j];

for (i=0; i<=12; i++)
    for (j=0; j<=time_max-1; j++)
        orig_chromo[order_of_evaluation[i+87]][j]=orig_chromo[order_of_evaluation[0]][j];
Appendix 12 - Toronto Hydro Electricity Rates

RESIDENTIAL RATE SCHEDULE - JANUARY 1, 1990

Energy Charges:
First 250 Kilowatt hours per month
- 10.27 cents per kWhr.
Balance of Consumption
- 6.90 cents per kWhr.
Minimum Monthly Bill
- $6.00

PROMPT PAYMENT DISCOUNT - 10%

COMMERCIAL - INDUSTRIAL RATE SCHEDULE - JANUARY 1, 1990

Monthly Demand Charge:
$5.11 per Kilowatt of the billing demand.

Energy Charge:
First 100 hours monthly use of the billing demand
- 11.11 cents per kWhr.
Next 100 hours monthly use of the billing demand
- 5.09 cents per kWhr.
Balance of the monthly consumption
- 3.52 cents per kWhr.
Minimum Monthly Bill
- $6.00

PROMPT PAYMENT DISCOUNTS - 10%
DISCOUNTS:

1. All rates are gross rates and subject to a prompt payment discount of 10% except where net rates are specifically stipulated.

2. Churches - 50% from the Commercial-Industrial Rate

3. Transformation Allowance for Supply at 4.16 KV - 1% plus $0.555 per KW per month of maximum demand discounted from the rate otherwise applicable.

4. Transformation Allowance for Supply at 13.8 KV - 1% plus $1.22 per KW per month of maximum demand discounted from the rate otherwise applicable.

5. Off-Peak Power (Commercial-Industrial)

For loads with an average billing demand for the months of April to September inclusive between 250 and 500 KW — 5%

For loads with an average billing demand for the months of April to September inclusive exceeding 500 KW — 10%

Off-Peak Power Overrun Penalty — $13.25 NET
RATES EFFECTIVE JANUARY 1, 1990

SERVICE TO CUSTOMERS WITH MAXIMUM MONTHLY DEMAND OF 2,500 KW TO 4,999 KW

The rates are gross rates and subject to a prompt payment discount of 10%. Where the customer is supplied at 4.16 kV or 13.8 kV, the appropriate transformation allowance shall also apply.

The Winter Rates shall be applicable to consumption during the months of January, February, March, October, November and December. The Summer Rates shall apply to consumption during the months of April, May, June, July, August and September.

1. For consumption in the Peak Period, which shall include the hours from 9:00 a.m. to 8:00 p.m., Monday to Friday, except public holidays:

Demand Charge per kilowatt of maximum demand per month occurring during the Peak Period:

NEW RATES
EFFECTIVE JAN. 1/90

<table>
<thead>
<tr>
<th></th>
<th>Winter</th>
<th>Summer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Charge:</td>
<td>$ 0.0691 per kWh</td>
<td>$ 0.0596 per kWh</td>
</tr>
</tbody>
</table>

2. For consumption in the Shoulder Period, which shall include the hours from 7:00 a.m. to 9:00 a.m. and 8:00 p.m. to 11:00 p.m., Monday to Friday, except public holidays:

NEW RATES
EFFECTIVE JAN. 1/90

Energy Charge:

<table>
<thead>
<tr>
<th></th>
<th>Winter</th>
<th>Summer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$ 0.0691 per kWh</td>
<td>$ 0.0596 per kWh</td>
</tr>
</tbody>
</table>

3. For consumption in the Off-Peak hours, which shall include all other hours not included in the Peak Period and the Shoulder Period:

NEW RATES
EFFECTIVE JAN. 1/90

Energy Charge:

<table>
<thead>
<tr>
<th></th>
<th>Winter</th>
<th>Summer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$ 0.0293 per kWh</td>
<td>$ 0.0219 per kWh</td>
</tr>
</tbody>
</table>
RATES EFFECTIVE JANUARY 1, 1990

SERVICE TO CUSTOMERS WITH
MAXIMUM MONTHLY DEMAND OF 5,000 KW OR GREATER

The rates are gross rates and subject to a prompt payment
discount of 10%. Where the customer is supplied at 4.16 kV or
13.8 kV, the appropriate transformation allowance shall also
apply.

The Winter Rates shall be applicable to consumption during the
months of January, February, March, October, November and
December. The Summer Rates shall apply to consumption during the
months of April, May, June, July, August and September.

1. For consumption in the Peak Period, which shall include the
hours from 9:00 a.m. to 8:00 p.m., Monday to Friday, except
public holidays:

Demand Charge per kilowatt of maximum demand per month
occurring during the Peak Period:

NEW RATES
EFFECTIVE JAN. 1/90

Winter - $ 9.86 per KW
Summer - $ 8.25 per KW

Energy Charge:
Winter - $ 0.0621 per kwhr
Summer - $ 0.0533 per kwhr

2. For consumption in the Shoulder Period, which shall include
the hours from 7:00 a.m. to 9:00 a.m. and 8:00 p.m. to 11:00
p.m., Monday to Friday, except public holidays:

NEW RATES
EFFECTIVE JAN. 1/90

Energy Charge:
Winter - $ 0.0621 per kwhr
Summer - $ 0.0533 per kwhr

3. For consumption in the Off-Peak hours, which shall include
all other hours not included in the Peak Period and the
Shoulder Period:

NEW RATES
EFFECTIVE JAN. 1/90

Energy Charge:
Winter - $ 0.0293 per kwhr
Summer - $ 0.0219 per kwhr
Appendix 13 - Input file bit-15.txt

Input file used for GAHOS-1A and GAHOS-1B

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0219</td>
<td>electricity cost for hour 0 [$/kW-hr]</td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td>electricity cost for hour 24 [$/kW-hr]</td>
</tr>
<tr>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>0.075</td>
<td></td>
</tr>
<tr>
<td>0.065</td>
<td></td>
</tr>
<tr>
<td>0.062</td>
<td></td>
</tr>
<tr>
<td>0.065</td>
<td></td>
</tr>
<tr>
<td>0.065</td>
<td></td>
</tr>
<tr>
<td>0.066</td>
<td></td>
</tr>
<tr>
<td>0.091</td>
<td></td>
</tr>
<tr>
<td>0.102</td>
<td></td>
</tr>
<tr>
<td>0.101</td>
<td></td>
</tr>
<tr>
<td>0.115</td>
<td></td>
</tr>
<tr>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>0.13</td>
<td></td>
</tr>
<tr>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>0.135</td>
<td></td>
</tr>
<tr>
<td>0.13</td>
<td></td>
</tr>
<tr>
<td>0.128</td>
<td></td>
</tr>
<tr>
<td>0.132</td>
<td></td>
</tr>
</tbody>
</table>
0.14
0.16
0.162
0.161
0.15
0.135

<---------------------- water demand rate for hour 24 [m³/s]
Appendix 14 - Input file test-15.txt

Input file used for GAHOS-2

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>electricity cost for hour 0 [$/kW·hr]</th>
<th>electricity cost for hour 24 [$/kW·hr]</th>
<th>water demand rate for hour 0 [m³/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.0219</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.08</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.075</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.065</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.062</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.065</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.065</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.066</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.091</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.102</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.101</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.115</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.12</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.13</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.14</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.135</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.13</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.128</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>0.132</td>
<td></td>
<td></td>
<td>-------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
0.14
0.16
0.162
0.161
0.15
0.135
<---------------------- water demand rate for hour 24 [m$^3$/s]

<table>
<thead>
<tr>
<th>Cost ($)</th>
<th>M&amp;O Cost %</th>
<th>Capacity (m$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>580000</td>
<td>0.070</td>
<td>2000</td>
</tr>
<tr>
<td>860000</td>
<td>0.069</td>
<td>4000</td>
</tr>
<tr>
<td>1280000</td>
<td>0.068</td>
<td>7000</td>
</tr>
<tr>
<td>1980000</td>
<td>0.067</td>
<td>12000</td>
</tr>
<tr>
<td>2820000</td>
<td>0.066</td>
<td>18000</td>
</tr>
</tbody>
</table>

Storage tank #1 (principal cost ($), M&O cost (%), Capacity (m$^3$))

<table>
<thead>
<tr>
<th>Pump Type #1</th>
<th>Cost ($)</th>
<th>M&amp;O Cost %</th>
<th>Flow (m$^3$/s)</th>
<th>Head (m)</th>
<th>Efficiency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1116090</td>
<td>0.10</td>
<td>0.191</td>
<td>60.3</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td>1023706</td>
<td>0.13</td>
<td>0.180</td>
<td>53.9</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>1178509</td>
<td>0.09</td>
<td>0.203</td>
<td>62.1</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td>1231594</td>
<td>0.08</td>
<td>0.231</td>
<td>55.3</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td>1193829</td>
<td>0.11</td>
<td>0.196</td>
<td>68.2</td>
<td>0.79</td>
<td></td>
</tr>
</tbody>
</table>

Pump type #1 (principal cost ($), M&O cost (%), flow (m$^3$/s), head (m), efficiency (%))

Pump type #5 (principal cost ($), M&O cost (%), flow (m$^3$/s), head (m), efficiency (%))
Appendix 15 - Output files showing the effect of the *absolute-best* function.

Output file of GAHOS-2 with the *absolute-best* function

Note: first column is the generation number, the second column is the current best solution,

<table>
<thead>
<tr>
<th>Generation</th>
<th>Best Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>151.028091</td>
</tr>
<tr>
<td>1</td>
<td>83.436676</td>
</tr>
<tr>
<td>2</td>
<td>80.940521</td>
</tr>
<tr>
<td>3</td>
<td>80.940521</td>
</tr>
<tr>
<td>4</td>
<td>80.940521</td>
</tr>
<tr>
<td>5</td>
<td>80.940521</td>
</tr>
<tr>
<td>6</td>
<td>80.940521</td>
</tr>
<tr>
<td>7</td>
<td>80.940521</td>
</tr>
<tr>
<td>8</td>
<td>79.236626</td>
</tr>
<tr>
<td>9</td>
<td>79.236626</td>
</tr>
<tr>
<td>10</td>
<td>79.236626</td>
</tr>
<tr>
<td>11</td>
<td>79.236626</td>
</tr>
<tr>
<td>12</td>
<td>79.236626</td>
</tr>
<tr>
<td>13</td>
<td>78.324989</td>
</tr>
<tr>
<td>14</td>
<td>77.532730</td>
</tr>
<tr>
<td>15</td>
<td>77.532730</td>
</tr>
<tr>
<td>16</td>
<td>77.532730</td>
</tr>
<tr>
<td>17</td>
<td>77.532730</td>
</tr>
<tr>
<td>18</td>
<td>77.532730</td>
</tr>
<tr>
<td>19</td>
<td>76.621094</td>
</tr>
<tr>
<td>20</td>
<td>76.621094</td>
</tr>
<tr>
<td>21</td>
<td>76.621094</td>
</tr>
<tr>
<td>22</td>
<td>75.828835</td>
</tr>
<tr>
<td>23</td>
<td>75.828835</td>
</tr>
<tr>
<td>24</td>
<td>75.828835</td>
</tr>
<tr>
<td>25</td>
<td>74.124939</td>
</tr>
<tr>
<td>26</td>
<td>74.124939</td>
</tr>
<tr>
<td>27</td>
<td>74.124939</td>
</tr>
<tr>
<td>28</td>
<td>74.124939</td>
</tr>
<tr>
<td>29</td>
<td>74.124939</td>
</tr>
<tr>
<td>30</td>
<td>74.124939</td>
</tr>
<tr>
<td>31</td>
<td>74.124939</td>
</tr>
<tr>
<td>32</td>
<td>74.124939</td>
</tr>
<tr>
<td>33</td>
<td>74.124939</td>
</tr>
<tr>
<td>34</td>
<td>72.421043</td>
</tr>
<tr>
<td>35</td>
<td>72.421043</td>
</tr>
<tr>
<td>36</td>
<td>72.421043</td>
</tr>
<tr>
<td>37</td>
<td>72.421043</td>
</tr>
<tr>
<td>38</td>
<td>72.421043</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>39</td>
<td>72.421043</td>
</tr>
<tr>
<td>40</td>
<td>72.421043</td>
</tr>
<tr>
<td>41</td>
<td>72.421043</td>
</tr>
<tr>
<td>42</td>
<td>72.421036</td>
</tr>
<tr>
<td>43</td>
<td>70.717148</td>
</tr>
<tr>
<td>44</td>
<td>69.805511</td>
</tr>
<tr>
<td>45</td>
<td>69.805511</td>
</tr>
<tr>
<td>46</td>
<td>69.013252</td>
</tr>
<tr>
<td>47</td>
<td>69.013252</td>
</tr>
<tr>
<td>48</td>
<td>69.013252</td>
</tr>
<tr>
<td>49</td>
<td>67.309357</td>
</tr>
<tr>
<td>50</td>
<td>67.309357</td>
</tr>
<tr>
<td>51</td>
<td>67.309357</td>
</tr>
<tr>
<td>52</td>
<td>67.309357</td>
</tr>
<tr>
<td>53</td>
<td>67.309357</td>
</tr>
<tr>
<td>54</td>
<td>67.309357</td>
</tr>
<tr>
<td>55</td>
<td>65.605461</td>
</tr>
<tr>
<td>56</td>
<td>65.605461</td>
</tr>
<tr>
<td>57</td>
<td>65.605461</td>
</tr>
<tr>
<td>58</td>
<td>65.605461</td>
</tr>
<tr>
<td>59</td>
<td>65.605461</td>
</tr>
<tr>
<td>60</td>
<td>65.605461</td>
</tr>
<tr>
<td>61</td>
<td>65.605453</td>
</tr>
<tr>
<td>62</td>
<td>65.605453</td>
</tr>
<tr>
<td>63</td>
<td>65.605453</td>
</tr>
<tr>
<td>64</td>
<td>65.605453</td>
</tr>
<tr>
<td>65</td>
<td>65.605453</td>
</tr>
<tr>
<td>66</td>
<td>65.605453</td>
</tr>
<tr>
<td>67</td>
<td>63.901566</td>
</tr>
<tr>
<td>68</td>
<td>63.901566</td>
</tr>
<tr>
<td>69</td>
<td>63.901566</td>
</tr>
<tr>
<td>70</td>
<td>63.901566</td>
</tr>
<tr>
<td>71</td>
<td>63.901566</td>
</tr>
<tr>
<td>72</td>
<td>63.901566</td>
</tr>
<tr>
<td>73</td>
<td>63.901566</td>
</tr>
<tr>
<td>74</td>
<td>63.901562</td>
</tr>
<tr>
<td>75</td>
<td>63.901562</td>
</tr>
<tr>
<td>76</td>
<td>63.901562</td>
</tr>
<tr>
<td>77</td>
<td>63.901562</td>
</tr>
<tr>
<td>78</td>
<td>63.901562</td>
</tr>
<tr>
<td>79</td>
<td>63.901562</td>
</tr>
<tr>
<td>80</td>
<td>63.901562</td>
</tr>
<tr>
<td>81</td>
<td>63.901562</td>
</tr>
<tr>
<td>82</td>
<td>63.901562</td>
</tr>
<tr>
<td>83</td>
<td>63.901562</td>
</tr>
<tr>
<td>84</td>
<td>63.901562</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
</tr>
<tr>
<td>85</td>
<td>63.901562</td>
</tr>
<tr>
<td>86</td>
<td>62.197666</td>
</tr>
<tr>
<td>87</td>
<td>62.197666</td>
</tr>
<tr>
<td>88</td>
<td>62.197666</td>
</tr>
<tr>
<td>89</td>
<td>62.197666</td>
</tr>
<tr>
<td>90</td>
<td>62.197666</td>
</tr>
<tr>
<td>91</td>
<td>62.197666</td>
</tr>
<tr>
<td>92</td>
<td>62.197666</td>
</tr>
<tr>
<td>93</td>
<td>62.197666</td>
</tr>
<tr>
<td>94</td>
<td>62.197666</td>
</tr>
<tr>
<td>95</td>
<td>62.197666</td>
</tr>
<tr>
<td>96</td>
<td>62.197666</td>
</tr>
<tr>
<td>97</td>
<td>62.197666</td>
</tr>
<tr>
<td>98</td>
<td>62.197666</td>
</tr>
<tr>
<td>99</td>
<td>62.197666</td>
</tr>
<tr>
<td>100</td>
<td>60.493771</td>
</tr>
<tr>
<td>101</td>
<td>60.493771</td>
</tr>
<tr>
<td>102</td>
<td>60.493771</td>
</tr>
<tr>
<td>103</td>
<td>60.493771</td>
</tr>
<tr>
<td>104</td>
<td>60.493771</td>
</tr>
<tr>
<td>105</td>
<td>60.493771</td>
</tr>
<tr>
<td>106</td>
<td>60.493771</td>
</tr>
<tr>
<td>107</td>
<td>60.493771</td>
</tr>
<tr>
<td>108</td>
<td>60.493771</td>
</tr>
<tr>
<td>109</td>
<td>60.493771</td>
</tr>
<tr>
<td>110</td>
<td>58.789871</td>
</tr>
<tr>
<td>111</td>
<td>58.789871</td>
</tr>
<tr>
<td>112</td>
<td>58.789871</td>
</tr>
<tr>
<td>113</td>
<td>58.789871</td>
</tr>
<tr>
<td>114</td>
<td>58.789871</td>
</tr>
<tr>
<td>115</td>
<td>58.789871</td>
</tr>
<tr>
<td>116</td>
<td>58.789871</td>
</tr>
<tr>
<td>117</td>
<td>57.085976</td>
</tr>
<tr>
<td>118</td>
<td>57.085976</td>
</tr>
<tr>
<td>119</td>
<td>57.085976</td>
</tr>
<tr>
<td>120</td>
<td>57.085976</td>
</tr>
<tr>
<td>121</td>
<td>57.085976</td>
</tr>
<tr>
<td>122</td>
<td>57.085976</td>
</tr>
<tr>
<td>123</td>
<td>57.085976</td>
</tr>
<tr>
<td>124</td>
<td>57.085976</td>
</tr>
<tr>
<td>125</td>
<td>57.085976</td>
</tr>
<tr>
<td>126</td>
<td>57.085976</td>
</tr>
<tr>
<td>127</td>
<td>57.085976</td>
</tr>
<tr>
<td>128</td>
<td>57.085976</td>
</tr>
<tr>
<td>129</td>
<td>57.085976</td>
</tr>
<tr>
<td>130</td>
<td>57.085976</td>
</tr>
</tbody>
</table>
Output file of GAHOS-2 without the *absolute-best* function

Note: first column is the generation number, the second column is the current best solution,

<table>
<thead>
<tr>
<th>Generation</th>
<th>Current Best Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>151.028091</td>
</tr>
<tr>
<td>1</td>
<td>82.644417</td>
</tr>
<tr>
<td>2</td>
<td>80.940521</td>
</tr>
<tr>
<td>3</td>
<td>79.236626</td>
</tr>
<tr>
<td>4</td>
<td>80.940521</td>
</tr>
<tr>
<td>5</td>
<td>80.940521</td>
</tr>
<tr>
<td>6</td>
<td>79.236626</td>
</tr>
<tr>
<td>7</td>
<td>80.940521</td>
</tr>
<tr>
<td>8</td>
<td>79.236626</td>
</tr>
<tr>
<td>9</td>
<td>79.236626</td>
</tr>
<tr>
<td>10</td>
<td>79.236626</td>
</tr>
<tr>
<td>11</td>
<td>77.532730</td>
</tr>
<tr>
<td>12</td>
<td>79.236626</td>
</tr>
<tr>
<td>13</td>
<td>77.532730</td>
</tr>
<tr>
<td>14</td>
<td>77.532730</td>
</tr>
<tr>
<td>15</td>
<td>77.532730</td>
</tr>
<tr>
<td>16</td>
<td>75.828835</td>
</tr>
<tr>
<td>17</td>
<td>77.532730</td>
</tr>
<tr>
<td>18</td>
<td>75.828835</td>
</tr>
<tr>
<td>19</td>
<td>74.917198</td>
</tr>
<tr>
<td>20</td>
<td>74.124939</td>
</tr>
<tr>
<td>21</td>
<td>74.124939</td>
</tr>
<tr>
<td>22</td>
<td>74.124939</td>
</tr>
<tr>
<td>23</td>
<td>74.124939</td>
</tr>
<tr>
<td>24</td>
<td>74.124939</td>
</tr>
<tr>
<td>25</td>
<td>74.124939</td>
</tr>
<tr>
<td>26</td>
<td>74.124939</td>
</tr>
<tr>
<td>27</td>
<td>72.421043</td>
</tr>
<tr>
<td>28</td>
<td>72.421043</td>
</tr>
<tr>
<td>29</td>
<td>72.421043</td>
</tr>
<tr>
<td>30</td>
<td>70.717140</td>
</tr>
<tr>
<td>31</td>
<td>70.717140</td>
</tr>
<tr>
<td>32</td>
<td>70.717140</td>
</tr>
<tr>
<td>33</td>
<td>70.717140</td>
</tr>
<tr>
<td>34</td>
<td>70.717140</td>
</tr>
<tr>
<td>35</td>
<td>70.717140</td>
</tr>
<tr>
<td>36</td>
<td>70.717140</td>
</tr>
<tr>
<td>37</td>
<td>70.717140</td>
</tr>
<tr>
<td>38</td>
<td>70.717140</td>
</tr>
<tr>
<td>39</td>
<td>69.013245</td>
</tr>
<tr>
<td>40</td>
<td>69.013245</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
</tr>
<tr>
<td>87</td>
<td>60.493767</td>
</tr>
<tr>
<td>88</td>
<td>60.493767</td>
</tr>
<tr>
<td>89</td>
<td>60.493767</td>
</tr>
<tr>
<td>90</td>
<td>60.493767</td>
</tr>
<tr>
<td>91</td>
<td>60.493767</td>
</tr>
<tr>
<td>92</td>
<td>60.493767</td>
</tr>
<tr>
<td>93</td>
<td>60.493767</td>
</tr>
<tr>
<td>94</td>
<td>60.493767</td>
</tr>
<tr>
<td>95</td>
<td>60.493767</td>
</tr>
<tr>
<td>96</td>
<td>60.493767</td>
</tr>
<tr>
<td>97</td>
<td>60.493767</td>
</tr>
<tr>
<td>98</td>
<td>60.493767</td>
</tr>
<tr>
<td>99</td>
<td>60.493767</td>
</tr>
<tr>
<td>100</td>
<td>58.789871</td>
</tr>
<tr>
<td>101</td>
<td>58.789871</td>
</tr>
<tr>
<td>102</td>
<td>58.789871</td>
</tr>
<tr>
<td>103</td>
<td>58.789871</td>
</tr>
<tr>
<td>104</td>
<td>58.789871</td>
</tr>
<tr>
<td>105</td>
<td>58.789871</td>
</tr>
<tr>
<td>106</td>
<td>58.789871</td>
</tr>
<tr>
<td>107</td>
<td>58.789871</td>
</tr>
<tr>
<td>108</td>
<td>58.789871</td>
</tr>
<tr>
<td>109</td>
<td>58.789871</td>
</tr>
<tr>
<td>110</td>
<td>58.789871</td>
</tr>
<tr>
<td>111</td>
<td>58.789871</td>
</tr>
<tr>
<td>112</td>
<td>58.789871</td>
</tr>
<tr>
<td>113</td>
<td>58.789871</td>
</tr>
<tr>
<td>114</td>
<td>58.789871</td>
</tr>
<tr>
<td>115</td>
<td>58.789871</td>
</tr>
<tr>
<td>116</td>
<td>58.789871</td>
</tr>
<tr>
<td>117</td>
<td>57.085976</td>
</tr>
<tr>
<td>118</td>
<td>57.085976</td>
</tr>
<tr>
<td>119</td>
<td>58.789871</td>
</tr>
<tr>
<td>120</td>
<td>57.085976</td>
</tr>
<tr>
<td>121</td>
<td>57.085976</td>
</tr>
<tr>
<td>122</td>
<td>58.789871</td>
</tr>
<tr>
<td>123</td>
<td>58.789871</td>
</tr>
<tr>
<td>124</td>
<td>57.085976</td>
</tr>
<tr>
<td>125</td>
<td>58.789871</td>
</tr>
<tr>
<td>126</td>
<td>58.789871</td>
</tr>
<tr>
<td>127</td>
<td>58.789871</td>
</tr>
<tr>
<td>128</td>
<td>58.789871</td>
</tr>
<tr>
<td>129</td>
<td>57.085976</td>
</tr>
<tr>
<td>130</td>
<td>57.085976</td>
</tr>
<tr>
<td>131</td>
<td>57.085976</td>
</tr>
<tr>
<td>132</td>
<td>57.085976</td>
</tr>
</tbody>
</table>