OPTIMAL MANAGEMENT OF COASTAL AQUIFERS USING HEURISTIC ALGORITHMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KORKUT DEMİRBAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CIVIL ENGINEERING

MARCH 2011

Approval of the thesis:

OPTIMAL MANAGEMENT OF COASTAL AQUIFERS USING HEURISTIC ALGORITHMS

submitted by **KORKUT DEMİRBAŞ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Civil Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Güney Özcebe _____
Head of Department, **Civil Engineering**

Assoc. Prof. Dr. A. Burcu Altan Sakarya _____
Supervisor, **Civil Engineering Dept., METU**

Prof. Dr. Halil Önder _____
Co-Supervisor, **Civil Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Metin Ger _____
İstanbul Aydın University

Assoc. Prof. Dr. A. Burcu Altan Sakarya _____
Civil Engineering Dept., METU

Prof. Dr. Can Elmar Balas _____
Civil Engineering Dept., Gazi University

Assoc. Prof. Dr. Nuray Tokyay _____
Civil Engineering Dept., METU

Assoc. Prof. Dr. Mehmet Ali Kökpınar _____
General Directorate of State Hydraulic Works

Date: 17.03.2011

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Korkut, Demirbaş

Signature :

ABSTRACT

OPTIMAL MANAGEMENT OF COASTAL AQUIFERS USING HEURISTIC ALGORITHMS

Demirbaş, Korkut

Ph.D., Department of Civil Engineering

Supervisor: Assoc. Prof. Dr. Ayşe Burcu Altan Sakarya

Co-Supervisor: Prof. Dr. Halil Önder

March 2011, 137 pages

Excessive pumping in coastal aquifers results in seawater intrusion where optimal and efficient planning is essential. In this study, numerical solution of single potential solution by Strack is combined with genetic algorithm (GA) to find the maximum extraction amount in a coastal aquifer. Seawater intrusion is tracked with the potential value at the extraction well locations. A code is developed by combining GA and a subroutine repeatedly calling MODFLOW as a numerical solver to calculate the potential distribution for different configurations of solution (trial solutions). Potential distributions are used to evaluate the fitness values for GA. The developed model is applied to a previous work by Mantoglou. Another heuristic method, simulated annealing (SA) is utilized to compare the results of GA. Different seawater prevention methods (i.e. injection wells, canals) and decision variables related to those methods (i.e. location of the injection wells or canals) are added to model to further prevent the seawater intrusion and improve the coastal aquifer benefit. A method called "Alternating Constraints Method" is introduced to improve the solution for the cases with variable location. The results show that both proposed method and the regular solution with GA or SA prove to be successful methods for the optimal management of coastal aquifers.

Keywords: Seawater Intrusion, Management of Coastal Aquifers, Genetic Algorithm, Simulated Annealing

ÖZ

KIYI AKİFERLERİNİN SEZGİSEL ALGORİTMALAR KULLANARAK OPTİMUM YÖNETİMİ

Demirbaş, Korkut
Doktora, İnşaat Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ayşe Burcu Altan Sakarya
Ortak Tez Yöneticisi: Prof. Dr. Halil Önder

Mart 2011, 137 sayfa

Kıyı akiferlerinden aşırı derecede su çekilmesi deniz suyu girişimine yol açmaktadır. Bu durum akiferlerde optimum (en uygun) ve etkin planlanma yapılmasını gerektirmektedir. Bu çalışmada Strack'ın tek potansiyel çözümününün sayısal çözümü, genetik algoritmayla birleştirilerek kıyı akiferlerindeki en yüksek su çekim miktarı bulunmaya çalışılmıştır. Tuzlu su girişimi, kuyu lokasyonlarındaki potansiyel değerlerine bakılarak takip edilmiştir. Genetik algoritma ile sayısal çözücü olarak MODFLOW'u tekrar tekrar çağıran bir altprogram birleştirilerek bir kod geliştirilmiş ve bu kod genetik algoritma tarafından ihtiyaç duyulan amaç fonksiyonu değerlerini hesaplamak üzere farklı çözüm konfigürasyonlarına karşılık gelen potansiyel dağılımını bulmak için kullanılmıştır. Geliştirilen model önceden Mantoglou tarafından çalışılan bir kıyı akiferi modeline uygulanmıştır. Genetik algoritma sonuçlarını karşılaştırmak üzere farklı bir sezgisel yöntem olan benzetilmiş tavlama yöntemi kullanılmıştır. Tuzlu su girişimini engellemek ve kıyı akiferden alınacak yararın artırılması amacıyla farklı tuzlu su önleme metodları ve bu metodlara yönelik karar değişkenleri modele eklenmiştir. Lokasyonların önemli olduğu işletme modelleri için sadece genetik algoritmanın kullanıldığı çözümü geliştirmek üzere "Değişken Kısıtlamalar Yöntemi" adında yeni bir metod geliştirilmiştir. Sonuçlar, önerilen yeni yöntemin ve sadece genetik algoritma ve benzetilmiş tavlama kullanılarak elde edilen yöntemin başarılı sonuçlar ortaya koyduğunu göstermiştir.

Anahtar Kelimeler: Tuzlu Su Girişi, Kıyı Akiferlerinin Yönetimi, Genetik Algoritma, Benzetimsel Tavlama

To my parents

ACKNOWLEDGMENTS

I would like to thank Assoc. Prof. Dr. Ayşe Burcu ALTAN SAKARYA for her invaluable support and suggestions during the study. This thesis would never have taken shape without her assistance. I would also like to thank Prof. Dr. Halil ÖNDER for his guidance and his careful reviews with a deep insight on the subject.

I would also like to express my most grateful thanks to Gizem KARSLI for her contributions and invaluable support in the turning points of this study. Our discussions always provided a different perspective, when things became complicated. Kubilay GÖNEN deserves mentioning for tirelessly evoking my interest and helping me not to lose my focus on the subject. Last but not the least; I am very grateful to my family for encouraging me in my studies. Without their support, this thesis would hardly be completed.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

LIST OF SYMBOLS

γ_s	Specific weight of seawater
γ_f	Specific weight of freshwater
h_s	Depth of interface below sea level
h_t	Elevation of water table above sea level
ρ_f	Density of fresh water
ρ_s	Density of seawater
δ	Constant that equals $(\rho_s - \rho_f)/\rho_f$
K	Hydraulic conductivity
b	Thickness of freshwater zone
h_f	Elevation of water table above datum
d	Elevation of sea level above datum
N	Incoming volumetric flow rate per unit horizontal area
W	Outgoing volumetric flow rate per unit horizontal area
z	Elevation of interface above datum
ϕ	Flow potential defined by Strack (1976)
ϕ_{toe}	Potential at the toe of seawater
m	Number of extraction wells
n	Number of injection wells
Q_{EXTi}	Extraction rate for i^{th} extraction well
Q_{INJj}	Injection rate for j^{th} injection well
R_{CAN}	Recharge rate for canal
α	Ratio of the economical value of injected water to the economical value of extracted water
β	Ratio of the economical value of recharged canal water to the economical value of extracted water
Δx	Length of the discretization in x-coordinate direction
Δy	Length of the discretization in y-coordinate direction
ϕ_i	Potential at i^{th} extraction well

Q_{EXTi}^{u}	Upper limit for i^{th} extraction well
Q_{EXTi}^{l}	Lower limits for i^{th} extraction well
Q_{INJj}^{u}	Upper limit for j^{th} injection well
Q_{INJj}^{l}	Lower limits for j^{th} injection well
x_{INJj}	x-coordinate of the i^{th} injection well
y_{INJj}	y-coordinate of the i^{th} injection well
x_{INJj}^{u}	Upper limit of x_{INJj}
x_{INJj}^{l}	Lower limit of x_{INJj}
y_{INJj}^{u}	Upper limit of y_{INJj}
y_{INJj}^{l}	Lower limit of y_{INJj}
x_{CAN}	x-coordinate of the canal
y_{CAN}	y-coordinate of the canal
x_{CAN}^{u}	Upper limit for x-coordinate of the canal
x_{CAN}^{l}	Lower limit for x-coordinate of the canal
y_{CAN}^{u}	Upper limit for y-coordinate of the canal
y_{CAN}^{l}	Lower limit for y-coordinate of the canal
l_{CAN}	Difference between the starting and ending coordinate of the canal in y-direction
l_{CAN}^{u}	Upper limit of l_{CAN}
l_{CAN}^{l}	Lower limits of l_{CAN}
c	Penalty constant
G	Maximum number of generation
g	Generation number
P	Population number
L	Total string length
η	Number of decision variables
l_i	String length of i^{th} decision variable
υ_i	Binary representation of i^{th} individual in a population

z_j	Binary representation of j^{th} decision variable
r_i	Base 10 value of i^{th} variable,
B_j^i	Value of j^{th} bit of the i^{th} variable
x_i	i^{th} decision variable
x_i^u	Upper bound of i^{th} decision variable
x_i^l	Lower bound of i^{th} decision variable
f_i	Fitness of the i^{th} individual
τ_i	Selection probability for the i^{th} individual,
f_{sum}	Sum of all fitnesses in a population
f_i^{scal}	Scaled fitness of i^{th} individual in a population
f_{avg}	Average fitness of individuals in a population
f_{avg}^{scal}	Average fitness of the scaled population
f_{best}^{scal}	Scaled fitness of the best individual.
λ_{scal}	Scaling constant
f_{best}	Best fitness in a population
p_c	Crossover probability that is selected between 0 and 1
p_m	Mutation probability that is selected between 0 and 1
p_m^i	Probability of mutation at i^{th} bit for uniform mutation.
f_{best}^*	Best fitness till current generation
f	Objective function
x_i^*	i^{th} decision variable of the best fitness till current generation
p_{acp}	Probability of acceptance of the new solution
f_{new}	Cost of the new solution for SA
f_{acp}	Cost of the previously accepted solution for SA
T	Control parameter for SA
X	Solution vector
X_0	Initial solution vector
X_{new}	New solution vector
X_{acp}	Accepted solution vector
X_{opt}	Optimal solution vector

x_i^{new}	New value of the i^{th} the decision variable for SA
R	A random positive real number between 0 and 1
v_i	Step length for the i^{th} decision variable
κ_i^{accp}	The number of accepted trials for each decision variable
ω	Constant parameter usually chosen between 1 and 2
N_s	Number of cycles till step-size adjustment
N_t	Number of cycles till T adjustment
t_{const}	Constant parameter used to update T
W_{EXTi}	i^{th} extraction well
CP_j	j^{th} control point
h_j^u	Upper limit for the hydraulic head at j^{th} control location
h_{CPj}	Head at j^{th} control point
W_{INJi}	i^{th} injection well

CHAPTER 1

INTRODUCTION

1.1 Overview

Rapidly growing human population and increasing human activity bring enormous pressure to the limited water supplies. Due to uncontrolled use, in many parts of the world water resources are in danger of running out.

Groundwater is the main source of freshwater in earth. High percentage of usable freshwater is stored under the surface, in the layers of soil formations (Leap, 2004). Of all these formations, aquifers are the most important ones, with their ability to store and transmit water with high efficiency. The water in the aquifers are supplied by the recharges from the higher grounds and with the potential energy developed it flows slowly through the soil pores feeding the surface water on the way till reaching to the sea at the end.

Coastal areas are one of the main centers of human settlements. The use of groundwater by extracting it from the wells has always been common and main source of water consumption, especially where surface water does not exist. Normally, groundwater flowing to sea is replenished by natural means. However, if it is consumed faster than it is replenished, freshwater storages will be filled with or contaminated by seawater and the valuable aquifers will be damaged at a point of no return or at least with a recovery cost of huge amounts.

The objective of the study is to introduce a management model for the maximum extraction of groundwater in a coastal aquifer while not permitting seawater intrusion. The proposed model includes the modeling of the coastal aquifer and combining with the global optimization method, Genetic Algorithm (GA). Simulation model is based on single potential solution by Strack (1976) which assumes that freshwater and saltwater are

immiscible and there exist a sharp interface separating the two fluids. The performance of the optimization algorithm will be tested on different groundwater scenarios including different seawater prevention methods (i.e. injection wells and canals). The result of the optimization algorithm GA will be compared with another heuristic method, Simulated Annealing (SA). To test the performance of GA and SA, Linear Programming (LP) or Mixed Integer Programming (MIP) will be utilized, whenever applicable.

1.2 Literature Survey

Under this title, a literature review based on previous studies on the subject is provided. Firstly, seawater intrusion modeling studies in coastal aquifers are presented. Then, management studies on coastal aquifers are summarized. Lastly, general studies on GA and SA are briefly discussed.

1.2.1 Seawater Intrusion Modeling

There are basically two different approaches for modeling seawater intrusion in coastal aquifers. In miscible fluid assumption, there exists a transition zone where seawater and freshwater mix due to hydrodynamic dispersion (Essaid, 1999). Miscible fluid assumption requires the simultaneous solution of the groundwater flow and mass transport equations. Analytical solutions to these equations are very limited and numerical solutions usually require high computation power (Essaid, 1990a). In the simpler approach, it is assumed that freshwater and saltwater does not mix and there exist a sharp interface that separates the two fluids. This assumption is valid especially when the width of the aquifer is narrow with respect to the depth of the aquifer (Essaid, 1986).

Sharp interface models are mainly classified into two (Essaid, 1990a): i)In a two-fluid approach, coupled equations of saltwater and freshwater flows are solved. ii)In a one-dynamic fluid approach, saltwater is assumed stagnant and only freshwater flow dynamics are utilized (equation regarding the freshwater flow is utilized). Seawater adopts the newly formed interface instantaneously.

Studies of seawater intrusion models based on sharp interface assumption goes back to the beginning of the second half of the twentieth century. Some of the major works include

Henry (1959), Shamir and Dagan (1971) and Mercer et al. (1980). Later studies enabled the application of sharp interface assumption on multilayer problems, including the works by Wilson and Costa (1982), Essaid (1986) and Huyakorn et al. (1996).

Henry (1959) proposed an analytical solution of the saltwater intrusion for one dimensional steady state flow. Seawater and freshwater are assumed immiscible. For different boundary conditions, the equations of the sharp interface are derived.

Shamir and Dagan (1971) introduced the partial differential equations that govern the motion of groundwater flow and describe the location of interface for a shallow unconfined aquifer. The equations are based on sharp interface and Dupuit-Forcheimer assumption (Bear and Zhou, 2004). Examples of numerical solution for a moving interface are given. It is reported that for a shallow aquifer, seepage length can be taken as zero without much error.

Mercer et al. (1980) presented a finite difference numerical model to solve the coupled equations of saltwater and freshwater flow. The assumptions include; sharp interface assumption, Dupuit-Forcheimer assumption and impermeable aquifer base.

Wilson and Costa (1982) solved finite element simulation of two layered coastal aquifer. Dupuit-Forcheimer assumption enabled the vertical integration of flow equations. The flow in vertical dimension is handled by adding leakage terms in governing equations.

Polo and Ramis (1983) presented a mathematical model to describe the saltwater freshwater motion with a sharp interface and Dupuit-Forchheimer assumption. Tests with analytical solutions are performed and reported to give good results. Numerical solution using finite difference approximation is also given.

Essaid (1986) introduced a quasi-three dimensional finite difference model to compare the two sharp interface models; coupled freshwater-saltwater model (two-fluid flow) and the Ghyben-Herzberg sharp interface model (one-dynamic fluid flow), (Essaid, 1999). He tested different cases to see the departure of one-dynamic fluid flow from the two-fluid flow. He concluded short-term responses and transitional responses between short term and long term can only be realistically simulated by including the dynamics of saltwater flow. In other

cases, where long term responses are required or short-term responses of an aquifer with high conductivity is studied, one fluid flow can be used.

Application of miscible flow for the aquifers with narrow transition zones are reported to arise some difficulties in the solution. Voss and Souza (1987) introduced some modifications for the variable density application of the simulation of coastal aquifers containing a narrow freshwater-saltwater transition zone.

In a study by Essaid (1990a), a finite difference model that simulates freshwater and saltwater flow separated by a sharp interface is developed to study a multilayered coastal aquifer. Flow between the layers is assumed vertical and added to the governing equations as a leakage term. Tip and toe locations for the interface are found by linearly extrapolating the discrete points (locations) found by numerical solver.

Galeati, et. al (1992) presented a numerical solution of the density dependent flow model for an unconfined aquifer. The proposed method is used to study the coastal aquifer in Italy, where seawater intrusion occurs due to excessive dewatering. Two dimensional flow in vertical cross-section is modeled using finite element method.

Huyakorn et al. (1996) developed a sharp interface numerical model to simulate saltwater intrusion in multilayered coastal aquifer systems. The dynamics of both freshwater and saltwater flow are considered.

Sakr (1999) studied the validity of sharp interface model in a confined coastal aquifer. Different cases were solved first with immiscible flow (sharp interface), then by density dependent model. To find the limitations of sharp interface, steady and non-steady models are solved by changing different parameter values; seepage factor, dispersion to advection ratio, geometry ratio and time scaling factor. Steady state simulations showed that the sharp interface approach is valid when the system is dominated by advection. The unsteady analysis showed that the use of sharp interface for all cases is sufficiently accurate at early times of simulation.

Karahanoğlu and Doyuran (2003) studied the case concerning the excavations below sea levels in a coastal aquifer in Kocaeli-Darıca, Turkey. Eleven new wells were drilled to

estimate the hydro-geological features and also for the monitoring of the area. Two different scenarios are considered to study the seawater intrusion into aquifer. Two dimensional finite element solution of the model is given.

Camur and Yazıcıgil (2005) introduced a three dimensional numerical density dependent flow and transport simulation model to predict the effects of an artificial water canal opening planned between Aegean Sea and historical Ephesus Site. The simulations included pre-pumping and pumping periods without canal and prediction period in the presence of canal. The results indicated that opening could cause further seawater intrusion and would affect the pumpage period for the nearby wells.

Ranjan et al. (2004) focused on the effects of geo-hydrological factors and recharge on saltwater-freshwater interface. A conceptual model based on sharp interface was considered to estimate the change in freshwater saltwater interface. The results showed that saltwater intrusion is far more sensitive to recharge than aquifer properties (i.e. storage coefficient, porosity, hydraulic conductivity).

Books about seawater intrusion includes complete books dedicated to the subject (i.e. Bear et al., 1999) and books with chapters that cover seawater intrusion concept (i.e. Delleur, 2004a)

Bear et al. (1999) edited a complete book about seawater intrusion, where different concepts of the subject are written by different authors. The book nicely summarizes the seawater intrusion subject, by including the mathematical models, analytical and numerical solution methods and case studies involving management problems. “Analytical Solutions” chapter by Cheng and Ouazar (1999) describes the derivation of single potential solution by Strack (1976) and different scenarios by changing the location and discharge value of an extraction well and a canal.

Delleur (2004a) gathered chapters written by different authors from the field in a book that covers the groundwater subject from fundamental mathematical theorems to different site applications. Chapters that involve seawater intrusion concept and management of seawater intrusion include “Elementary Groundwater Flow and Transport Processes” by Delleur (2004b) and “Seawater Intrusion into Coastal Aquifers” by Bear and Zhou (2004).

1.2.2 Management of Coastal Aquifers

Initial efforts to support and improve the operation of groundwater systems by simulation and optimization techniques are started in early 1970's (Benhachmi et al., 2003). Gorelick (1983) made a review of the single objective linear simulation management models for groundwater. According to Gorelick, there are basically two types of management models; embedded and response matrix approach. In the embedded approach, formulations of groundwater equations are directly written into optimization formulation. In response matrix approach, a matrix of influence coefficients is formed by utilizing the unit response of the system to a unit impulse. The method which is valid for linear systems requires iterations for nonlinear systems (Tokgöz et al., 2002 and Demirbaş, 2003).

Later, Ahfeld and Heidari (1994) summarized the characteristics of hydraulic control design problems and approaches to solve these problems in a groundwater system. Field applications are classified into water supply management problems and remediation management problems.

Management of coastal aquifers usually require the simulation of the equations related to seawater and freshwater flow. The studies can be classified according to the type of the interface assumption for the simulation (see Section 1.2.1). Among the many studies that utilizes the sharp interface assumption, studies made by Willis and Liu (1984), Shamir et al. (1984), Finney et al. (1992), Willis and Finney (1998), and Emch and Yeh (1998) can be listed. Studies that utilize density dependent flow are relatively new. However, the number of applications is increasing with the increasing power of today's computers. Some of the works are Das and Datta (1999, 2001).

Constraining the saltwater intrusion is often provided by indirect manners like constraining drawdown at control points or minimizing seawater flow at certain locations (Benhachmi et al., 2003). Others include tracking the seawater encroachment or checking the saltwater concentration at certain locations (i.e. extracted water).

The management model applications in saltwater intrusion often require the use of nonlinear optimization due to complexity of governing equations (Emch and Yeh, 1998). Later using

global optimizers like GA became popular. Some of the works are Cheng et al. (2000), Park et al. (2003), Park and Aral (2003), Mantoglou et al. (2004), Benhachmi et al. (2001, 2003), Katsifarakis and Petala (2004, 2006), Mantoglou and Papantoniou (2004, 2008), Qahman and Larabi (2005). Bhattacharjya and Datta (2005). Studies made by SA also exist, although their numbers are few. Rao et al. (2003) and Rao et al. (2004) can be listed as examples.

Willis and Liu (1984) proposed a multiobjective management model, which is applied to Yun Lin Basin in Taiwan. For the basin, water demands has become larger than the natural recharge values which results in saltwater intrusion. A finite element model is formed to simulate the aquifer response. Conflicting objectives include maximizing the groundwater extraction to supply the water demand and groundwater levels in the coastal region where seawater intrusion occurs.

Shamir et al. (1984) made a study to determine the optimal annual operation of a costal aquifer by considering four objectives; a desired groundwater surface map, desired location of the seawater toe, desired concentration map and the minimization of the energy (cost) for pumping. A trade-off curve is formed to choose between different desirable solutions. The model is applied to a coastal aquifer in Israel.

Finney et al. (1992) used sharp interface assumption and coupled equations of freshwater and saltwater flow for the simulation of multi-layered aquifer. The flow in aquifer layers are based on flow equations in two dimensions. Flows in vertical direction between different layers are provided with the recharge terms in governing equations. A multi-objective optimization model is developed in order to satisfy a certain water demand and to minimize the seawater volume. The resulting problem is solved with a nonlinear optimization solver package, MINOS (Murtagh and Saunders, 1993). Seawater intrusion is constrained with restrictions on possible pumping and recharge locations.

Hallaji and Yazıcıgil (1996) used response matrix approach for the combined simulation optimization of seawater intrusion model in southern Turkey. In their solution, they used head constraint in specified locations for the control of saltwater intrusion. Several management models are studied to find the optimal operating policy. Tradeoff curves are built for the objectives of optimal pumping rates and minimum pumping costs. The best

alternative found is explained as pumping excess water from the most productive wells and transport them to the remaining areas.

Emch and Yeh (1998) used multi-objective management model to manage the groundwater resources and together with the coastal surface water. Two fluid sharp interface assumption is utilized. The conflicting objectives considered were the minimization of seawater intrusion and minimization of cost of supplying water. Saltwater intrusion is tracked with the level of interface. Water demand is given as a lower bound constraint. Additional constraints related to minimum drawdowns at control points are also included. MINOS used as an optimization tool and SHARP (Essaid, 1990b) as the flow simulator.

Two fluid sharp interface assumptions is utilized by Willis and Finney (1998) for the management of a coastal basin in Taiwan. The basin is modeled using finite difference approximation. Resulting management problem is solved using two different nonlinear optimization methods. Different objective functions are combined in a single potential function by using different weighing constants for each objective function term. Objectives include minimization of pumping, minimization of injection costs and minimization of seawater intrusion. Seawater intrusion is tracked with the position of the toe of the interface.

Das and Datta (1999) utilized embedding technique where numerically approximated simulation equation of the flow is incorporated as constraint set of optimization model for the management of multiobjective management of a coastal aquifer. Seawater, freshwater interaction is modelled as a density dependent miscible flow and transport model. Nonlinear optimization solver package MINOS, is used as an optimization tool.

Three dimensional density dependent miscible flow and transport model for a seawater intruded coastal aquifer is studied by Das and Datta (2001). Effects of vertical recharge, boundary conditions and location of pumping on simulation model are investigated and following conclusions are derived: i) amount of seawater intrusion is found inversely proportional with recharge, ii) seawater intrusion increased with decreasing boundary head. iii) Increased pumping increased salinity. iv) Location of the well has a significant effect on seawater intrusion (i.e. away from the coast, there is less danger of saltwater intrusion). Lastly, Das and Datta (2001) experimented with a series of barrier wells and concluded that

installing series of barrier wells that are close to the sea boundary is an effective way to avoid seawater intrusion.

Zhou et al. (2002) utilized response matrix approach to manage the freshwater sources without permitting seawater intrusion on a multilayered aquifer system in Leizhou Peninsula in southern China. A quasi-three dimensional flow (horizontal flow, flow in the vertical dimension are included in governing equations by leakage terms) on a finite element model is used to simulate groundwater levels in the aquifer system. Control of the seawater intrusion is attained by restricting the water levels at points along the coast.

Motz et al. (2004) used response matrix approach to combine two dimensional miscible flow and a linear optimization model. Response matrices are formed by repeatedly calling groundwater flow and transport model SUTRA (Voss, 1984).The model is used for the management of seawater intrusion in Göksu Delta in Southern Turkey.

One of the early applications of genetic algorithm on groundwater problems is the study by McKinney and Lin (1994). Three different management problems are solved to test the performance of GA on groundwater management. These problems include a pumping management problem which maximizes the extraction amount by constraining drawdown at certain control points, a cost management problem which minimizes the installation and operating costs for a number of potential well locations to provide a certain water demand and a cost minimization for an aquifer remediation problem. Results showed that GA can effectively and efficiently be used to obtain global (or at least near global) optimal solutions to these groundwater management problems.

Aly and Peralta (1999) introduced a simulation and optimization approach for single and multi-objective planning period for a groundwater contamination remediation. Proposed optimization model was tested with GA and a nonlinear optimization algorithm (mixed integer nonlinear programming) using different scenarios. GA found better results than the mixed integer nonlinear programming for more complex problems.

Cheng et al. (2000) used the analytical solutions of the freshwater flow to analyze the cases with one-well, two-well and one-well with recharge canal. For multiple wells, a management problem is developed to optimize the maximum pumpage and GA is used to search for the

optimal solution. Sharp interface is assumed for the transition zone. Dupuit-Forchheimer assumption is used to vertically integrate the flow equation. Together with the above assumption and assumption of static seawater lead to the single potential solution by Strack (1976). Seawater intrusion is tracked with the potential taking a specific value at the well location. First, critical pumpage rates for a single well intruded with saltwater is studied. From the derived equations, a design chart is provided, where intrusion distance and maximum pumping rate versus toe potential is given. The same equations are derived for two well case where the wells are separated at a distance from each other and at some distance away from the coast. A dimensionless chart is presented where maximum discharge is given as a function of toe potential and the ratio of the distance between the wells to the distance to the coast. For the case with recharge canal and one extraction well, equations are provided for the changing locations of the canal. From the results, it is seen that critical extraction is increasing when canal moves away from the coast till a critical distance. From this distance on, the critical extraction rate remains the same.

Analytical solutions for a two dimensional steady saltwater-intruded coastal unconfined aquifer based on the sharp interface assumption and Strack (1976) solution is successfully applied to a management model using genetic algorithm by Benhachmi et al. (2001).

Benhachmi et al. (2003) applied GA to a seawater intruded groundwater management problem in Miami coast. For the simulation model, analytical solution of the single potential formulation by Strack (1976) is used. Aquifer geometry is simplified (i.e. straight coastline, constant thickness, homogeneity, etc.) to enable the use of analytical solution. Solution of the aquifer for multiple wells is found by using method of images as given by Strack (1976). The management objectives were to maximize the economic benefit from pumped water and minimize the utility cost of lifting the water. Toe encroachment invading the wells (i.e. if the toe location from the coast in front of a well is greater than the distance of the well from the coast) are included in the objective function as a penalty to control the seawater intrusion for the management.

Park and Aral (2003) combined single potential formulation by Strack and GA to optimize the conflicting objectives of maximum amount of extraction and minimum distance to the coast without letting seawater intrusion. They concluded that including the well locations as a decision variable improved the results significantly.

For the large number of the simulation evaluations required for the solution by GA, Park et al. (2003) utilized a PC cluster of 32 processors. Fitness function for GA is computed in a parallel manner in order to decrease the solution time. Coupled equations of freshwater and saltwater flow with sharp interface approach are used for the solution of the simulation model. Seawater intrusion is constrained by groundwater levels at certain points, the quantity of saltwater pumped and the salt concentrations at the extraction wells. These constraints are added to the objective function as a penalty term with different weighting constants. Objective was to withdraw maximum amount of groundwater.

In the study made by Katsifarakis and Petala (2004, 2006), the numerical evaluation of groundwater flow with boundary element code and seawater inflow is checked with the sign of flow rate value at the coastal boundary element. Optimal location and optimal withdrawal rate for two wells is studied with the proposed model.

Mantoglou (2003) developed a model that seeks for the optimal pumping rates for a coastal aquifer. Single potential solution based on sharp interface assumption and Ghyben-Herzberg relation is used for the simulation model. Seawater intrusion is tracked by the toe location. Analytical solutions are compared with numerical solutions. The methodology was applied to an aquifer in Greek island of Kalymnos.

Mantaglou et al. (2004) compared a nonlinear optimization algorithm (sequential quadratic programming) and evolutionary algorithm for the maximum extraction rates in a coastal aquifer using the single potential solution by Strack (1976). They found that although sequential quadratic programming reaches solution far more quickly, it could stuck on local optimums.

Mantoglou and Papantoniou (2004) used evolutionary algorithms to manage a pumping network in a coastal aquifer. The formulation of the constraints is based on numerical formulation of sharp interface assumption and Ghyben-Herzberg approximation that leads to single potential formulation of Strack (1976). The objectives were to optimize the total pumping rates and well locations. The proposed methodology was applied to a Greek island. Evolutionary algorithm was developed using MATLAB and differential equation for flow

(single potential solution for an unconfined aquifer) is solved by using the finite difference solver, MODFLOW (McDonald and Harbourgh, 1988).

Bhattacharjya and Datta (2005) utilized density dependent miscible flow and transport model and GA to maximize the water extracted in a coastal aquifer. Saltwater intrusion is controlled by constraining the salt concentration in extracted water. To reduce the computational burden due to high number of function evaluations, a trained artificial neural network (approximate simulator) instead of the simulation model is used to calculate the response of aquifer to different solution configurations.

Qahman and Larabi (2005) considered a hypothetical coastal confined aquifer, which is a modified version of Henry's problem (Voss, 1984). The flow was modeled as three dimensional density dependent miscible flow and transport model. Simulation scenarios include maximizing the total volume of extracted water, maximizing the profit of selling water, minimizing the operational and water treatment costs and minimizing the salt concentration of pumped water. Maximum allowable salt concentration in extracted water and minimum head in the wells are used as constraint, for seawater intrusion. GA was used as the optimization tool.

Mantoglou and Papantoniou (2008) introduced a method, where pumping locations are optimized by GA where for each individual extraction rates are optimized with linear programming. The single potential solution by Strack (1976) is used to find the response of the aquifer to different discharge combinations.

SA applications in groundwater management are relatively few. In a study made by Kuo et al (1991), optimal locations and discharge values for the wells in a pump and treat system in a groundwater contamination site is solved by using SA. The algorithm is designed as combinatorial algorithm and decision variables are chosen accordingly (i.e. from a specified set of potential values). Although required more computational time, SA algorithm is reported to give better results for which the search space is non-smooth due to well installation cost involved in objective function.

Wang and Zheng (1998) compared the performance of SA and GA for the solution of a groundwater management problem, where either maximum groundwater demand or

minimum cost of extraction is searched for. It is reported that GA and SA gave identical or better results than the linear and nonlinear programming. SA is reported to require less number of function evaluations. However, Wang and Zheng (1998) also stated that SA solutions are highly dependent on empirical parameters.

Cunha (2002) compared four different optimization methods, including simulated annealing for groundwater development planning for a hypothetical problem designed for a real aquifer. The model to be solved included capital costs for the placement (drilling and installing costs) and variable costs for the operation of the new wells. Cunha (2002) reported that all the solutions except simulated annealing is very sensitive to the initial solution. Using different initial solutions, simulated annealing showed its robustness by almost always achieving the same solutions. Though the solutions are reported to be much more time consuming than the others, results by SA found to be better.

Rao et al. (2003) used simulated annealing to search for the optimal groundwater solution in deltaic regions without inducing excessive saltwater intrusion. SA was coupled with SHARP flow model and was used to find the optimal location and pumpages of extraction wells. The computational burden is lowered by replacing SHARP with an artificial neural network. SHARP flow model is a solver based on sharp interface assumption and coupled equations of freshwater and saltwater flow. Seawater intrusion is controlled by a interface elevation constraint at specified nodes.

Rao et al. (2004) controlled the seawater intrusion through a series of barrier wells (wells that extracts the saltwater and throws it back to the sea) while maximizing the extraction amount. Multi-objective management problem is solved using SA as an optimizer and SEAWAT (Guo and Langevin, 2002) for simulating the seawater groundwater dynamics based on density dependent groundwater flow and transport model. Seawater intrusion is controlled by checking the salt concentration in the wells and restraining head values at control points. To reduce the computational burden arised from the solution of density dependent problem, SEAWAT model is replaced with a trained artificial neural network. Simulated annealing algorithm is utilized to solve the optimization problem. Tradeoff curves are formed between two objectives, maximization of groundwater development through production wells and minimizing the pumpage from the barrier wells.

In an effort to identify the groundwater source parameters (i.e. pumping source location, pumping rate and period, etc.), Lin and Yeh (2008) used SA as an optimization tool and MODFLOW as the numerical solver. Following the results, they gave least number of parameter values to analyze the source information. It is reported that combination of SA and MODFLOW gave successful results including the cases with measured errors.

1.2.3 Optimization Methods (GA and SA)

There are various books in literature written about GA. Most of these books are introductory books that cover the subject by explaining the GA from the scratch. Among those Coley (1999), Haupt and Haupt (2004) and Sivanandam and Deepa (2007) can be listed. Others took the subject on a broader perspective by covering the family of algorithms involving GA (i.e. evolutionary algorithms). Examples are the books by Michalewicks (1996) and De Jong (2006). Majority of these books discuss how GAs work by using the empirical evidence. However, there are few giving the theoretical foundation like the famous work by Goldberg (1989).

Since SA is a stochastic algorithm that is built over natural phenomena like GA, there are discussion about SA in most of the above books. However, the scope of these discussions is rather limited. Books that cover SA in a complete book also exist, like the study by Van Laarhoven and Aarts (1987). In following paragraphs, a detailed surveying of these books will be given.

De Jong (2006) studied the family of evolutionary algorithms, which include GA as a subarea. Evolutionary algorithm is defined as the iteration of population of individuals evolving to a better fitness with certain selection, reproduction and mutation mechanisms. De Jong (2006) then described GAs as the type of evolutionary algorithm where all the individuals are replaced by the offsprings after one generation and mating individuals are selected according to their fitness for which the offsprings are generated by recombining generic information by crossover and random alteration of the bits by mutation. To better understand the basics of evolutionary algorithm, De Jong (2006) returned to the roots of evolutionary system in biology which mainly consists of; i) population or populations competing for a limited source ii) constantly changing population with the births of new individuals and death of old individuals iii) fitness which is defined as the degree of survival

for a specific individual iv) reproduction which gives offsprings similar to parents but somehow different (concept of inheritance).

Banzhaf and Rieves (1999) edited and published the proceedings of workshop, named "Foundations of Genetic Algorithm". The workshop is organized under the direction of International Society for Genetic Algorithm and reviewed with well-known authors from the field. The workshop titles mostly focused on the theoretical issues related to GA and improvements in GA.

Sivanandam and Deepa (2007) presented the basic concepts related to GA, with special emphasis on the roots of GA, which are connected to biology. This was inspiring since the first application of GA by Holland (1975) was for the simulation of the evolution of a system in nature. Some biological background on cells, genes, chromosome and DNA are given. The structure of genetic information on DNA (genotype), what it represents (phenotype) and how it is carried to offsprings by reproduction is discussed in an effort to understand the counterparts of these concepts in GA.

In his well-known book, Goldberg (1989) tried to explain why and how GAs work both with empirical and theoretical work. "Schema Theorem" introduced by Holland (1975) is explained in detail. In the end, case studies are discussed, where special attention is given to machine learning.

In his book, Coley (1999) made a brief introduction to GA, where basic and advanced operators and how they affect the results of the algorithm are discussed. Different methods to improve the simple GA (i.e. using hybrid methods, advanced coding techniques, etc.) are explained. A whole chapter is dedicated to how to write a basic GA algorithm and a sample code written in Fortran 90 is given in the end.

Michalewics (1996) listed the various optimization areas where GA is successfully applied, including the hydraulic control problems. He defined GAs as: "During the last decade, the significance of optimization has grown even further, many important large-scale combinatorial optimization problems and highly constrained engineering problems can only be solved approximately on present day computers. Genetic algorithms aim at such complex problems. They belong to the class of probabilistic algorithms, yet they are very different

from random algorithms as they combine elements of directed and stochastic search. Because of this, GAs are also more robust than existing directed search methods. Another important property of such genetic based search methods is that they maintain a population of potential solutions - all other methods process a single point of the search space. Examples on combinatorial optimization and the ones which include continuous decision variables are discussed." For binary representation, "Schema Theorem" by Golderg (1989) is given as a theoretical foundation of why GAs work. Floating number representation is compared to binary coded representation and from the results obtained, it is concluded that floating number representation is superior to binary representation.

Different types of GA operators (i.e. one point, two point, uniform crossover etc), and their effects on searching the solution space is given in detail by Haupt and Haupt (2004). Concepts are explained with basic applications and methods to handle advanced applications are listed. The other algorithms based on natural phenomena including SA are summarized. Classification of different types of GAs and their definitions are given in a chapter.

Van Laarhoven and Aarts (1987), defined SA as an approximation algorithm that can be applicable to a wide variety of problem. Theoretical proof that SA will always converge to a global optimum is given by using the theory of Markov chains (Markov chains are sequence of algorithms which define the acceptance criteria for iterative solutions (See Section 2.2.2)). However, since any implementation of SA is an approximation of the assumptions accepted for the mentioned proof (i.e. infinite number of Markov chains), there is no guarantee that a global optimum will be found using SA. In the book, different types of SA implementations are discussed and performance for these cases are analyzed. Van Laarhoven and Aarts (1987) reported that adaptive SAs (i.e. SAs with adaptive parameters that changes during the run) performed better than SAs with constant parameters. Beside classical structure designed for combinatorial problems, SAs that use continuous decision variables are also covered.

1.2.3.1 GA Parameters

Studies on finding the optimal parameter set for GA started from the point where GA started to be used as a function optimizer (De Jong, 1992). Goldberg (1989) summarized the extensive study and conclusions of De Jong (1975) on optimal GA parameters. De Jong tested five different optimization functions with different population numbers, crossover

constants and mutation probabilities. He defined two performance measures for testing the success of GA; offline and online performance. Online performance is defined as the average of the fitness values till current generation while online performance is defined as the average of the best fitness till current generation. From the results, he concluded that small populations improve initial performance while high populations improve long time performance. He tested populations ranging from 50 to 200. He also concluded that a crossover rate of 0.6 is a good balance between online and offline performance. De Jong also suggested low mutation rates that are inversely proportional to population number. Mutation rates larger than 0.1 are reported to converge the algorithm to random selection.

Grefenstette (1986) tested different GA parameters on five different test functions. Based on five independent runs, best GA parameters for online performance was reported as population number = 30, crossover rate = 0.95 and mutation rate = 0.01 while scaling and elitism is on. Best offline performance was reported when population number = 80, crossover rate = 0.45 and mutation rate = 0.01 while scaling is on but no elitism. Grefenstette (1986) also suggested crossover rates of 1.0 performing better if the stochastic effects are reduced by inducing more selection pressure.

Schaffer et al. (1989) experimented with 8400 different combinations on crossover, mutation rate, population number and different types of GA operators. The result of each combination was averaged over 10 independent runs. Best performance was achieved when two point crossover is selected and population number = 20-30, crossover rate = 0.75-0.95 and mutation rate = 0.005-0.01.

Haupt and Haupt (2004) made 200 independent runs for 21 different population numbers and 21 different mutation rates. Average number of calls needed to arrive to an acceptable solution is accepted as a measure of success. The best combinations for the 5 different functions ranged from 8 to 88 for the population, 0.01 to 0.41 for the mutation rate. Haupt and Haupt (2004) used 12 bits for the encoding of each decision variable.

Haupt and Haupt (2004) also compared the binary and floating-point representation on an optimization problem on the design of antenna array. Population number of 20, mutation rate of 0.2 were used. To reduce the impact of variation, the algorithm was repeated and averaged over 10 independent runs. The results of binary representation outperformed the results of

floating-point representation. They reported that success of binary representation may be due to different sizes of the search space for two different representation (10^{21} potential solutions for binary representation with 21 variables and an encoding on base 10, ∞ potential solutions for floating representation)

Michalewicz (1996) utilized the idea of variable population number and tested four functions for the comparison of fixed and variable population number. Initial population number, crossover and mutation rate are taken as 20, 0.65 and 0.015, respectively. The length of the chromosomes was 20. Twenty independent runs were performed for each instance and measures of performance and fitness were averaged over twenty runs. He concluded that variable population size gives better performance in the expense of higher evaluation numbers. He found 75, 15, 75 and 100 as the optimal population number for the four functions.

Michalewicz (1996) also studied the effect of binary and floating point representation. The average values obtained from ten independent runs are presented. Population number was 60 and the number of generations was set to 20,000. Binary representation used 30 bits for representing each decision variable; for 45 variables, which adds up to 1350 bits. Best results were almost identical for the two representations although solutions with floating-point representation converged to near optimal results faster.

Back (1993) focused their studies on mutation operator and suggested a mutation rate of about $1/L$ (one over total string length). Back and Schutz (1996) later introduced a method on variable mutation rate which starts with a higher rate of mutation (i.e. 0.5) and decreases to $1/L$ as the algorithm progress.

1.3 Research Objectives

The general goal of this study is to develop a management model in order to optimize the maximum benefit in a coastal aquifer where there is a threat of seawater intrusion. The model will be modified to examine different seawater intrusion prevention methods (i.e injection wells, canals, etc.) with the emphasis on efficient and/or optimal use of coastal aquifer.

For the optimization process, a heuristic algorithm, GA will be used. GAs are accepted to be a robust optimization techniques that are applicable to wide range of problems. However, their robustness sometimes limits their efficiency. GAs are reported to be less effective than tailored techniques which already have information about the search space. For improved performance, GA parameters should be modified according to problem in hand. GA operators should also be chosen accordingly.

GA has difficulty to find near optimal results when domain (search space) gets larger (i.e. dense discretization) and decision variables or constraints increase. Including advanced operators in GA and improving the algorithm helps to find near optimal results. Improved methods are expected to be developed as more complex problems occur. To enable easier modification for the aforementioned processes, a GA code will be developed.

The results will be compared with the results of another widely used heuristic algorithm, SA. To examine the efficiency of GA, SA and improved GA; LP or MIP will be utilized, whenever applicable. It is expected that comparative results will give the user idea of utilizing different management alternatives for a convenient coastal aquifer model.

1.4 Organisation of the Thesis

The thesis is organized in six chapters. Chapter 1 is an introduction to thesis which include a brief overview of the study, a literature survey and the current subchapter. Chapter 2 describes the simulation and management model used in the thesis. Management model includes the formulations related to optimization methods, GA and SA. Chapter 3 summarizes the program code, explaining the operators of GA, command by command. The fourth chapter contains the solution of a sample problem by GA. It explains how GAs work by including detailed explanation of GA operators and their implementation on a groundwater dewatering example. The chapter also includes the parameter optimization for GA using the same dewatering problem. The parameters found are utilized for the rest of the study. Chapter 5 includes the application and discussion of the results. Different scenarios (problems) by including different seawater prevention methods are tested on the same simulation model using the simulation optimization methods developed. Each problem is followed by the corresponding results and discussions. Final chapter summarizes the conclusions.

CHAPTER 2

BACKGROUND

2.1 Simulation Model

In a coastal aquifer, since freshwater is less dense than saltwater, freshwater floats over the saltwater, where two fluids meet. The seawater and freshwater mix due to hydrodynamic dispersion and a finite thickness of a brackish water interface is formed. In the simpler approach, this transition zone is assumed narrow and there exist a sharp interface that separates two fluids. This way, transition zone is neglected. However, flow dynamics still depends both on saltwater and freshwater. To further simplify the solution, Ghyben and Herzberg (independently in 1888 by Ghyben and 1901 by Herzberg) assumed static equilibrium where seawater is stagnant and there is hydrostatic pressure distribution in the freshwater region (Bear and Zhou, 2004). This is equal to a dynamic equilibrium where freshwater flows horizontally (Dupuit-Forcheimer assumption) and seawater is stagnant. This way flow dynamics of saltwater can be ignored and only flow related to freshwater gives the solution. The interface position is found by the Ghyben-Herzberg relation. Hydrostatic pressure distribution in both zones leads to a seepage through a single point. It is assumed that seawater adopts the freshwater shape instantaneously. (See Figure 2.1).

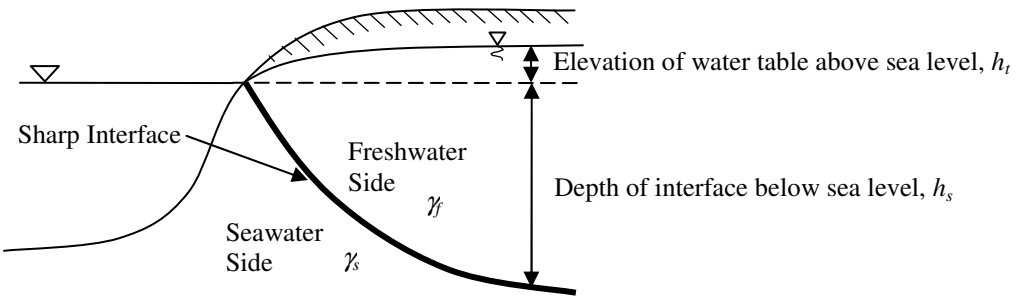


Figure 2.1 Ghyben-Herzberg interface model

The pressure at a point on the interface is the same whether approached from the freshwater side or the saltwater side. Thus,

$$\gamma_s h_s = \gamma_f (h_s + h_t) \tag{2.1}$$

where,

γ_s is the specific weight of seawater,

γ_f is the specific weight of freshwater,

h_s is the depth of interface below sea level,

h_t is elevation of water table above sea level.

Solving for h_s we get,

$$h_s = \frac{\gamma_f}{\gamma_s - \gamma_f} h_t = \frac{\rho_f}{\rho_s - \rho_f} h_t = \frac{1}{\delta} h_t \tag{2.2}$$

where,

ρ_f is the density of freshwater,

ρ_s is the density of seawater,

δ is $(\rho_s - \rho_f)/\rho_f$.

Substitution of ρ_f =1000 kg/m^3 and ρ_s = 1025 kg/m^3 in Eqn. 2.2, yields commonly known equation, $h_s = 40h_t$, since δ =0.025. According to the above model, the depth of interface below sea level is 40 times the elevation of water table above sea level. That means for a steady model, if there is 1 unit of decline in the groundwater head, there will be 40 units of increase in the depth of interface till it reaches a steady state. The relation given by Eqn. 2.2 is called the law of Ghyben and Herzberg (Bear and Zhou, 2004).

Sharp interface approach based on Ghyben-Herzberg approximation is more appropriate for modeling long term responses of freshwater zones or short term responses in aquifers where saltwater can move in and out easily (Essaid, 1999). Figure 2.2 shows the vertical cross-section of such an aquifer. Toe location is tip of the interface; the farthest point where seawater reaches through freshwater. Toe location divides the aquifer into two zones. Zone 1 is the freshwater only zone and Zone 2 is where freshwater flows over stagnant seawater

(Mantoglou, 2003). In both zones, freshwater flows horizontally, in other words, Dupuit-Forcheimer assumption is valid. For the above conditions, aquifer storativity is ignored and so governing equation becomes time independent.

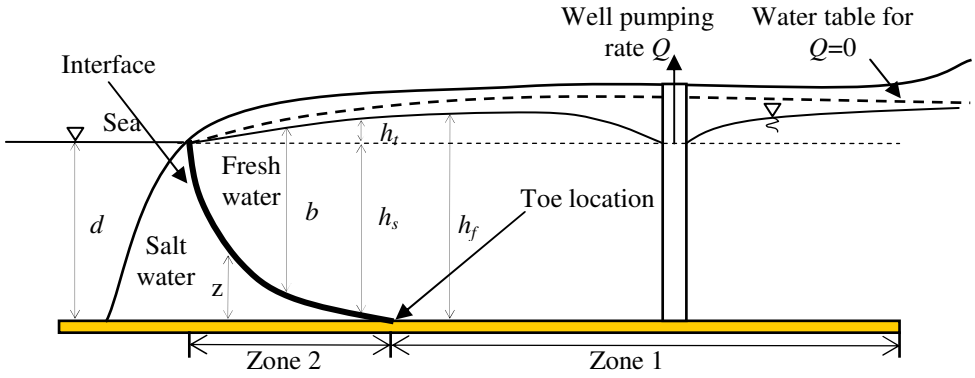


Figure 2.2 Coastal unconfined aquifer pumped by one well

The governing equation of steady groundwater flow for Zone 1 and Zone 2 is expressed as follows;

Zone 1:

$$\frac{\partial}{\partial x}\left(Kb\frac{\partial h_f}{\partial x}\right)+\frac{\partial}{\partial y}\left(Kb\frac{\partial h_f}{\partial y}\right)+N-W=0 \tag{2.3}$$

in which, $b=h_f$

Zone 2:

$$\frac{\partial}{\partial x}\left(Kb\frac{\partial h_f}{\partial x}\right)+\frac{\partial}{\partial y}\left(Kb\frac{\partial h_f}{\partial y}\right)+N-W=0 \tag{2.4}$$

in which, $b=h_f-d+h_s$

In Eqns. 2.3 and 2.4,

K is the hydraulic conductivity,

b is the thickness of freshwater zone,

h_f is the elevation of water table above datum,

d is the elevation of sea level above datum,

N is the incoming volumetric flow rate per unit horizontal area (recharge rate),
W is the outgoing volumetric flow rate per unit horizontal area (withdrawal rate),
z is the elevation of interface above datum.

Using Eqn. 2.2,

$$\delta h_s = h_f - d \tag{2.5}$$

Strack (1976) defined flow potential for the two zones in order to express Eqns. 2.3 and 2.4 as a single function.

Zone 1:

$$\phi = \frac{1}{2}\left[h_f^2 - (1+\delta)d^2\right] \tag{2.6}$$

Zone 2:

$$\phi = \frac{1+\delta}{2\delta}\left(h_f - d\right)^2 \tag{2.7}$$

ϕ function is continuous and smooth across the zones and satisfies the following equation (Strack,1976).

$$\frac{\partial}{\partial x}\left(K\frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(K\frac{\partial \phi}{\partial y}\right) + N - Q = 0 \tag{2.8}$$

Toe location may be found either from Eqn. 2.6 or 2.7 by letting $h_f - d = \delta h_s$ (Eqn. 2.5) and $h_s = d$, which is equal to;

$$\phi_{toe} = \frac{(1+\delta)\delta}{2} d^2 \tag{2.9}$$

where, ϕ_{toe} is the potential at the toe of seawater.

Toe location represents the encroachment of the interface, the farthest point till seawater has progressed. As the freshwater discharged to the sea is reduced (i.e. with increasing extraction demand), freshwater head above seawater reduces and seawater wedge moves toward inland. Further increase in amount of extraction leads to pumping saline water from the wells.

Analytical solution of Eqn. 2.8 is limited, except for the cases with simple boundary conditions and/or with limited number of inputs (like wells, recharge, etc.). Numerical solution methods on the other hand, give the flexibility of solving wide range of problems with the required precision in an adequate time.

The single potential formulation given above (Eqn. 2.8) is in fact the governing equation for a confined aquifer with unit thickness. If the boundary conditions are known, it can be transformed to a problem that can be solved numerically with a finite difference solver, such as MODFLOW (Harbaugh and McDonald, 1996). After solving for $\phi(x,y)$, elevation of water table, h_f and depth of seawater-freshwater interface, h_s can be estimated with the following functions.

Zone 1:

$$h_f = (2\phi + (1+\delta)d^2)^{\frac{1}{2}} \text{ and } h_s = d \qquad \text{For } \frac{(1+\delta)\delta}{2} d^2 \le \phi \qquad (2.10)$$

Zone 2:

$$h_f = (\frac{2\delta\phi}{1+\delta})^{\frac{1}{2}} + d \text{ and } h_s = \frac{1}{\delta} \times (\frac{2\delta\phi}{1+\delta})^{\frac{1}{2}} \qquad \text{for } 0 \le \phi \le \frac{(1+\delta)\delta}{2} d^2 \qquad (2.11)$$

Before presenting the management formulation, it may be convenient to discuss the early upconing event seen in the solutions of single potential solution. This event was also mentioned by Cheng et al. (2000) and Cheng and Ouazar (1999) (See Section 1.2.1). Figure 2.3 shows the advancement of seawater wedge as the discharge value at the wells in the example are slowly increased. Figure 2.3-a shows the interface just before $\phi = \phi_{toe}$ at the well nearest to the coast. Discharge value of the wells for this case is 738 m^3/day. There is no upconing for this situation. When the discharge values are increased till $\phi > \phi_{toe}$ at the well nearest to the coast (i.e. to a discharge value of 740 m^3/day), an upconing is formed under the well (Figure 2.3-b). This is not possible since upconing has no connection with seawater. However, once the discharge values are further increased by less than 7% (i.e. to 782 m^3/day), seawater wedge reaches the well location (Figure 2.3-c). Taking ϕ_{toe} at the well locations as a constraint for seawater intrusion for all the above cases makes us on the safe side, since seawater wedge have not reached the wells yet. Figure 2.3-d shows the response of the aquifer, when discharges are further increased to 1000 m^3/day. Once, wells starts to pump seawater, the solution by single potential solution does not reflect the flow dynamics (Cheng et al., 2000).

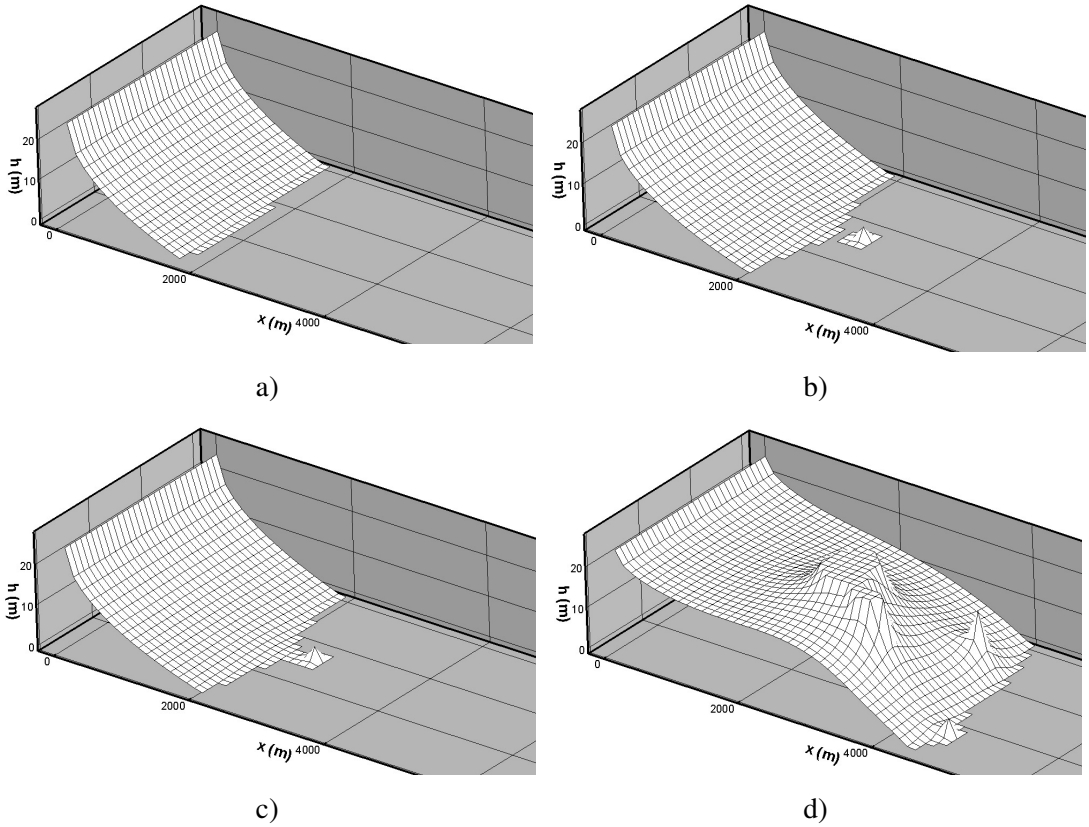


Figure 2.3 Interface elevation for different discharge values; a) Q_{EXTi}=738 m^3/day b) Q_{EXTi}=740 m^3/day c) Q_{EXTi} =782 m^3/day d) Q_{EXTi} =1000 m^3/day for i=1,2…5

2.2 Management Model

Using the simulation model, it is possible to get the assessment of how the aquifer behaves under different scenarios and compare them in an effort to attain a better solution. However, it is usually hard to cover all the alternatives, which will fulfill the required constraints. In many cases, a better solution might have easily been omitted.

Transforming the simulation model to an optimization problem, by defining the objectives and including the constraints, give the user the freedom to search for the optimal solution using various optimization methods. Using optimization in a management model not only enables finding the optimal solutions in an adequate time but also enhances the flexibility of adopting similar management objectives to the model.

The objective in this study is to find the maximum benefit in a coastal aquifer without letting seawater intrusion into the extraction wells. ϕ_{toe} represents the border where the seawater intrusion has reached. If the ϕ values at the extraction wells are smaller than ϕ_{toe}, it is assumed that wells are intruded by seawater.

The general mathematical formulation for the optimization problem in this study is as follows;

The objective function,

$$Maximize\ f = \sum_{i=1}^{m} Q_{EXTi} - \alpha \sum_{j=1}^{n} Q_{INJj} - \beta.\Delta x.\Delta y.l_{CAN}.R_{CAN} \tag{2.12}$$

Set of constraints,

$$\phi_i \geq \phi_{toe} \qquad \text{for } i=1,2\ldots.m \tag{2.13}$$

$$Q^l_{EXTi} \leq Q_{EXTi} \leq Q^u_{EXTi} \qquad \text{for } i=1,2\ldots.m \tag{2.14}$$

$$Q^l_{INJi} \leq Q_{INJj} \leq Q^u_{INJj} \qquad \text{for } j=1,2\ldots.n \tag{2.15}$$

$$R^l_{CAN} \leq R_{CAN} \leq R^u_{CAN} \tag{2.16}$$

$$x^l_{INJj} \leq x_{INJj} \leq x^u_{INJj} \qquad \text{for } j=1,2\ldots.n \tag{2.17}$$

$$y^l_{INJj} \leq y_{INJj} \leq y^u_{INJj} \qquad \text{for } j=1,2\ldots.n \tag{2.18}$$

$$x^l_{CAN} \leq x_{CAN} \leq x^u_{CAN} \tag{2.19}$$

$$y^l_{CAN} \leq y_{CAN} \leq y^u_{CAN} \tag{2.20}$$

$$l^l_{CAN} \leq l_{CAN} \leq l^u_{CAN} \tag{2.21}$$

where; m is the number of extraction wells and n is the number of injection wells. Q_{EXTi} is the extraction rate for i^{th} extraction well and Q_{INJj} is injection rate for j^{th} injection well. R_{CAN} is the recharge rate for the canal (rate of water recharged to the canal). α is the ratio of the economical value of injected water to the economical value of extracted water. β is the ratio of the economical value of recharged canal water to the economical value of extracted water. Δx is the length of the discretization in x-coordinate direction and Δy is the length of

constant discretization in y-coordinate direction. ϕ_i is the potential at i^{th} extraction well. Q^u_{EXTi} and Q^l_{EXTi} are the upper and lower limits for i^{th} extraction wells, Q^u_{INJj} and Q^l_{INJj} are the upper and lower limits for j^{th} injection wells and R^u_{CAN} and R^l_{CAN} are the upper and lower limits for the canal, respectively. x_{INJj} is the x-coordinate of the i^{th} injection well with x^u_{INJj} and x^l_{INJj}, the upper and lower limits of x_{INJj}, respectively. y_{INJj} is the y-coordinate of the i^{th} injection well with y^u_{INJj} and y^l_{INJj}, the upper and lower limits of y_{INJj}, respectively. x_{CAN} is x-coordinate of the canal. x^u_{CAN} and x^l_{CAN} are the upper and lower limits for the x-coordinate of the canal, respectively. y_{CAN} is y-coordinate of the canal. y^u_{CAN} and y^l_{CAN} are the upper and lower limits for y-coordinate, respectively. The canal runs parallel to the coast and l_{CAN} is defined as the difference between the starting and ending coordinate of the canal in y direction. The upper and lower limits of l_{CAN} are expressed by l^u_{CAN} and l^l_{CAN}, respectively. A sketch showing the geometrical elements of the management problem is given in Figure 2.4.

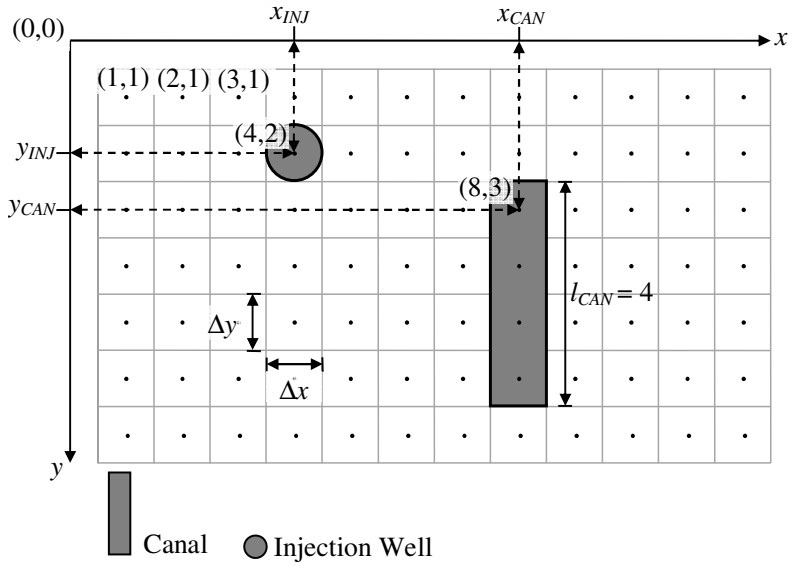


Figure 2.4 A sketch of the geometrical elements of the management problem

Eqns. 2.12 to 2.21 is a general formulation of all the elements that will be covered in this thesis. Not all of them will be included at the same time in a single optimization case. For instance, in an optimization problem that includes extraction elements only, the second and

third terms of right hand side of Eqn. 2.12 and Eqns. from 2.15 to 2.21 will drop from the optimization formulation.

Since objective function and the constraints are linear functions of decision variables (i.e. discharge rates), the problem (Eqns. 2.12 to 2.21) is a linear optimization problem. ϕ values show linear response to stress and response matrix approach can be used to transform the management problem into LP or MIP (MIP is used if the constraints related to locations (Eqns. 2.17 to 2.21) are involved in the management problem). Resulting LP or MIP, then can be solved by using a linear optimization code (Demirbas, 2003).

In this study, the solution to the defined management problem is found by using heuristic algorithm methods, GA and SA. The results, whenever applicable, are compared with the results of LP or MIP.

For the solution with LP and MIP, MODMAN (Greenwald, 1998) is utilized. MODMAN repeatedly calls MODFLOW to assemble the response matrix coefficients. MODFLOW provides the numerical solution of ϕ distribution, for which the governing equations are identical to a hypothetical confined aquifer. Finally, the optimization code LINDO (Lindo Systems) is used for the solution of LP or MIP.

The defined optimization problem is a constrained problem. In order to deal with constraints in GA, a number of methods are reported in literature. Davis and Steenstrup (1987) summarized three of these methods; imposing great penalties for constraint violation, imposing moderate penalties, and creating decoders that creates only feasible solutions. Davis and Steenstrup (1987) reported that imposing great penalties on individuals that violate constraints omits the infeasible solutions from the optimization process and if the search space contains many individuals violating the constraints, then GA will spend most of its time on these individuals and if any feasible solution is found it may dominate the population resulting in a premature convergence. Davis and Steenstrup also reported if the last method is used, the resulting problem will be very computer intensive and also hard to implement.

In this study, penalty method is used to convert the constrained problem into an unconstrained problem and moderate penalties (i.e. increasing the penalty till no constraint violation in the optimum solution) are used for constraint violations. Since the upper and lower bounds of the decision variables (Eqns. 2.14 to 2.21) are entered as input file to the genetic code, only constraints related to state variables (Eqn. 2.13) are added to the objective function as penalty term. The penalty is proportional to the degree of violation of the constraints.

Unconstrained objective function,

$$Maximize\ f = \sum_{i=1}^{m} Q_{EXTi} - \alpha \sum_{j=1}^{n} Q_{INJj} - \beta.\Delta x.\Delta y.l_{CAN}.R_{CAN} - c\sum_{i=1}^{m}(\max(0, \phi_{toe} - \phi_i))^2 \qquad (2.22)$$

where,

c is a problem dependent penalty constant.

2.2.1 Genetic Algorithm

GAs are powerful search and optimization technique for optimal solutions when conventional techniques are not adequate. They are successful in finding optimal or near optimal solutions for highly convex and nonlinear problems (Goldberg, 1989). In literature, increasing number of applications exist in wide range of areas. General mechanisms are based on biological evolution by natural selection, which was first discovered by Charles Darwin in 1851. The algorithm as known today is first developed by Holland (1975). Although initial application is used for the simulation of an adaptive system, not long time passed since GAs are begun to be utilized as function optimizers. In GA, decision variables are coded as strings of binary digits which are called “chromosome” or “individual”. Each set of strings (individuals) corresponds to a solution in the problem and each of those individuals has a degree of success according to the objective function, which is called fitness. Population composed of many individuals evolves under specified selection rules that maximize their fitness.

Mainly three mechanisms; selection, crossover and mutation drives the new generations to the optimal or near optimal solution. Selection is done based on the fitness of the chromosomes which may be defined as the rank of the objective function.

Flow chart of simple GA (Figure 2.5) can be given as;

1. A number of individuals (population number) are randomly generated as initial population.

2. Fitness of each individual is evaluated. Fitness is the numerical representation of objective function plus a degree of penalty if any constraint is violated. In selection of the new generation, the individuals with higher fitness will have more chance to be selected as mating individuals. Sometimes an elitist strategy is utilized; where the individual with highest fitness is directly pass to the next iteration.

3. After mating individuals are selected, whether they will undergo crossover or not is decided with a probability. Random crossover locations on each string pairs are chosen. Then, information is exchanged between two strings starting from the selected location (One-point crossover).

4. Mutation operator is considered for each bit of the string with a chosen probability. If the probability occurs, that specific bit is randomly flipped. While crossover exchange the information that is already in genetic pool, mutation operator may add new information to genetic diversity. Mutation is generally accepted as an insurance policy against the premature loss of chromosomes in generations (Goldberg 1989).

5. Step 2 to 5 is repeated for a number of generations till a convergence criteria is met (i.e. till maximum number of generations, G is reached). Number of current generation is denoted by g. The individual with the best fitness is the optimal or near optimal solution.

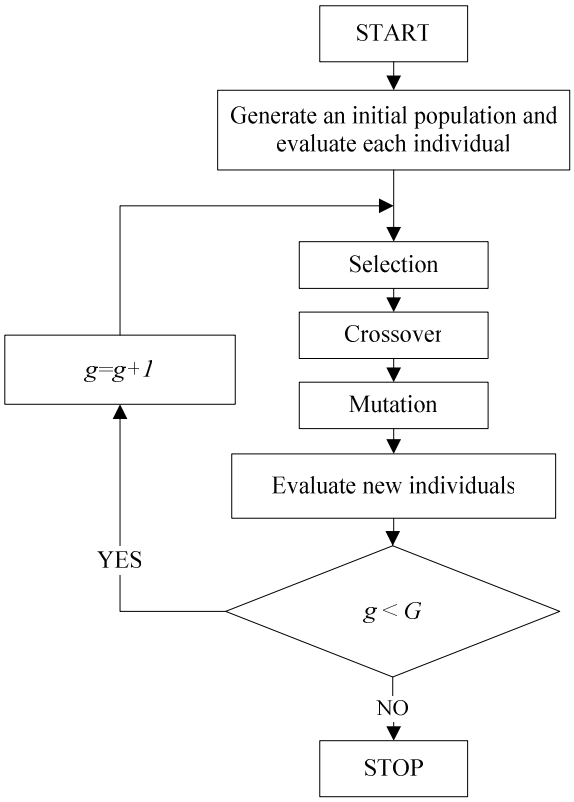


Figure 2.5 Flow chart of simple genetic algorithm

GAs usually require several objective function evaluations. If the objective function requires the solution of a numerical model, then with dense discretization and numerous input variables, obtaining near optimal solutions can take quiet some time. In order to achieve moderate results in reasonable time, modifications should be made in the model and/or number of function evaluations should be decreased by improving the algorithm (Coley, 1999).

In the following sections, equations of initialization of the algorithm, how to handle problems with more than one decision variable, encoding and decoding of the solution vectors will be covered. Then, selection, crossover and mutation types will be explained with samples. Elitism, which is a mechanism to guarantee the selection of the best individual for the next generation will be discussed.

2.2.1.1 Initialization of the Algorithm

GA starts with the random generation of the binary coded individuals. The number of individuals in a population is called population number and is denoted by *P*. Each individual represents a specific set of decision variables string length of which is decided by the user. Total string length is simply found by Eqn. 2.23 (Michalewicz, 1996).

$$L = \sum_{i=1}^{\eta} l_i \tag{2.23}$$

where,

L is the total string length of individual,

η is the number of decision variables,

l_i is the string length of i^{th} decision variable.

The binary representation of an individual is the concatenation of the binary representations of each decision variables (Eqn. 2.24).

$$\upsilon_i = z_1 z_2 \ldots\ldots z_\eta \tag{2.24}$$

where,

υ_i is the binary representation of i^{th} individual in a population,

z_j is the binary representation of j^{th} decision variable (j=1,2…...η).

Table 2.1 An example of an initially generated population with P=5, η =2, l_1=6, l_2=4 and L=10 = (l_1+ l_2)

Binary Representation of the First Decision Variable	Binary Representation of the Second DecisionVariable	Binary Representation of the Individual
101110	1001	1011101001
101110	1000	1011101000
010011	0110	0100110110
000100	1010	0001001010
110100	0111	1101000111

2.2.1.2 Selection and Decoding of the Variables

To evaluate the success of a solution that an individual represent, set of decision variables for the specific individual should be decoded from their binary form and used in estimation of the objective function. Binary coded decision variables are first converted to their base 10 values and then converted to their real values. The base 10 value of the variables are calculated by Eqn. 2.25.

$$r_i = \sum_{j=1}^{l_i} B_j^i . 2^{j-1} \tag{2.25}$$

where,

r_i is the base 10 value of i^{th} variable,

B_j^i is the value of j^{th} bit of the i^{th} variable.

The real values of the individuals are calculated by using the transformation formulation given by Eqn. 2.26. The value depends both on boundary values and the base 10 value of the encoded variables.

$$x_i = x_i^l + r_i . \frac{(x_i^u - x_i^l)}{2^{l_i} - 1} \tag{2.26}$$

where,

x_i is the i^{th} decision variable,

x_i^l is the lower bound for variable x_i,

x_i^u is the upper bound for variable x_i.

When all set of variables are decoded to their real values, objective function for each individual in the population can be evaluated by using Eqn. 2.27.

$$f_i = f(x_1, x_1, ..., x_\eta) \qquad i=1.....P \tag{2.27}$$

where,

f_i is the fitness of the i^{th} individual.

Two types of selection are covered. First, the equations of fitness proportionate selection will be given and linear scaling, which is an improvement for fitness proportionate selection, will be explained. Then, a different selection mechanism, tournament selection will be described.

2.2.1.2.1 Fitness Proportionate Selection

Mating individuals are selected in probabilities proportionate to their fitnesses. The chance of selection of an individual is equal to the ratio of the fitness of that individual to the sum of all fitnesses in the population (Eqn. 2.28).

$$\tau_i = f_i / f_{sum} = f_i / \sum_{i=1}^{P} f_i \tag{2.28}$$

where,

τ_i is the selection probability for the i^{th} individual,

f_{sum} is the sum of all fitnesses in a population.

During selection, better than average individuals will get exponentially more chance (chance to be selected more than once), average individuals will get even chance and worse than average individuals will have less chance to be selected for the next generation (Goldberg, 1989).

Linear Scaling

Early in a GA run, few individuals with very high fitness with respect to others will have much more chance to be selected. These individuals can easily dominate the population resulting a premature convergence. Also later in optimization, when the fitness values do not show much variation, the algorithm may have difficulties in selecting between good and better individuals. Instead, selection process approaches to random selection. Scaling the fitness of population by pivoting them about the average population fitness may help for both situations (Goldberg, 1989).

For linear scaling;

$$f_i^{scal} = a.f_i + b \tag{2.29}$$

where,

f_i^{scal} is the scaled fitness of i^{th} individual in a population.

It is assumed that average fitness of the population remains unchanged after scaling;

$$f_{avg}^{scal} = f_{avg} \tag{2.30}$$

where,

f_{avg} is the average fitness of the population,

f_{avg}^{scal} is the average fitness of the scaled population.

The scaled fitness of the best individual is assumed to be average fitness times scaling constant;

$$f_{best}^{scal} = \lambda_{scal}.f_{avg} \tag{2.31}$$

where,

f_{best}^{scal} is the scaled fitness of the best individual,

λ_{scal} is the scaling constant.

From Eqn. 2.29,

$$b = f_i^{scal} - a.f_i \tag{2.32}$$

If Eqn. 2.32 is added up for all the individuals in the population (i.e. i=1,2.....*P*), the following equation is obtained;

$$P.b = \sum_{i=1}^{P} f_i^{scal} - a.\sum_{i=1}^{p} f_i \tag{2.33}$$

and

$$b = \frac{\sum_{i=1}^{P} f_i^{scal}}{P} - \frac{a.\sum_{i=1}^{p} f_i}{p} \tag{2.34}$$

Using Eqn. 2.30, Eqn. 2.34 becomes

$$b = f_{avg} - a.f_{avg} = (1-a).f_{avg} \tag{2.35}$$

Inserting Eqn. 2.35 into Eqn. 2.29 gives

$$f_i^{scal} = a.f_i + (1-a).f_{avg} \tag{2.36}$$

For the best fitness in a population, f_{best} the above equation takes the form

$$f_{best}^{scal} = a.f_{best} + (1-a).f_{avg} \tag{2.37}$$

Then inserting Eqn. 2.31 into 2.37 becomes

$$\lambda_{scal}.f_{avg} = a.f_{best} + (1-a).f_{avg} \tag{2.38}$$

and

$$a = \frac{(\lambda_{scal} - 1) f_{avg}}{f_{best} - f_{avg}} \tag{2.39}$$

Inserting Eqns. 2.35 and 2.39 into Eqn. 2.29, fitness of all the individuals can be scaled according to the following equation.

$$f_i^{scal} = \frac{(\lambda_{scal} - 1) f_{avg}}{f_{best} - f_{avg}} f_i + (1-a).f_{avg} \tag{2.40}$$

Negative values may arise after scaling if very small fitness values with respect to the average fitness population occur (Goldberg, 1989). Fitness values should be positive in order that operators function properly (i.e. selection). In this study, negative values after scaling are set to zero.

2.2.1.2.2 Tournament Selection

In tournament selection, two or higher number of individuals are selected randomly and the individual with best fitness is selected for mating. Tournament selection is reported to be more effective when fitness surface does not show much variation (Coley, 1999). In this study, number of individuals in a tournament is taken as two.

Both fitness proportionate and tournament selection is repeated till required number of individuals are selected for mating. At the end of selection, some individuals may have chance to be selected more than once and some may have no chance to be selected at all.

2.2.1.3 Elitism

In fitness proportionate selection, best individual in a population has the highest probability to be selected for mating (Eqn. 2.28). However due to stochastic effects, there is no guarantee for it to be selected both for fitness proportionate selection and tournament selection. Sometimes in GA, elitist strategy is utilized, so that the best individual of current population is carried directly to the next generation. Usual application is not to apply

mutation or crossover to the best individual (Goldberg, 1989). Instead, individual with best fitness takes the place of the individual with the minimum fitness in new generated population. Elitist strategy is reported to improve efficiency of algorithm for some problems. However, in some cases it can put much pressure on selection with the result of early convergence to a local optimum (Coley, 1999).

After the selection process, the mates are passed to crossover and mutation, in turn. With the completion of these two processes, first generation of GA is complete.

2.2.1.4 Crossover

Whether crossover will be applied to chosen mates or not is decided by the crossover constant, p_c. p_c is usually a real number decided between 0 and 1. Three different types of crossover is utilized; one point, two point and uniform crossover.

One point crossover

The point of crossover is also randomly chosen. The bits after the chosen point are interchanged between the mates. If there is no crossover, the mates are left as is. An example is given in Table 2.2 where the bits after 4th bit (shown in bold) are interchanged.

Table 2.2 One point crossover example

	Binary Coding of Mating Individuals
Before Crossover	1010**1011010011** 1111**0010111010**
After Crossover	1010**0010111010** 1111**1011010011**

Two point crossover

Two random points on individuals are chosen. The bits between those points are interchanged. The other bits are left as is. An example is given in Table 2.3 where the bits between 3th bit and 11th bit (shown in bold) are interchanged.

Table 2.3 Two point crossover example

	Binary Coding of Mating Individuals
Before Crossover	101**01010110**011 111**10011101**1010
After Crossover	101**10011101**0011 111**01010110**1010

Uniform Crossover

In uniform crossover, whether a bit is interchanged between mating individuals is decided randomly. Uniform crossover is accepted as the general form of point crossovers. An example is given in Table 2.4 where interchanged bits are shown in bold.

Table 2.4 Uniform crossover example

	Binary Coding of Mating Individuals
Before Crossover	1**0**1**0**1**0**1**1**1**0**1**0**0**1**1 1**1**1**1**0**0**1**0**1**1**1**1**0**1**0
After Crossover	1**1**1**0**0**0**1**0**1**1**1**0**0**1**1 1**0**1**1**1**0**1**1**1**0**1**1**0**1**0

2.2.1.5 Mutation

The next step of GA after crossover is mutation. Mated individuals (whether they undergo crossover or not) are then passed to mutation process. For each bit, it is decided whether there will be a mutation or not. If mutation occurs, the value of the bit is simply flipped. The probability of mutation is set with mutation number, p_m which is decided betwen 0 and 1. An example to mutation is given in Table 2.5, where the bits that undergo mutation are shown in bold.

Table 2.5 Mutation operator example

	Binary Coding of Individual
Before Mutation	1010**0**0100001011110**0**0001000000
After Mutation	1010**1**0100001011110**1**0001000000

Uniform Mutation:

The probability of mutation for each bit in an individual is same for normal mutation. If mutation occurs in lower bits, the change in the value of individual is small relative to a mutation in higher bits. Normal mutation can be adjusted so that lower bits will have more chance of mutation while higher bits will have less chance (Eqn. 2.41).

$$p_m^i = p_m / 2^i \tag{2.41}$$

where,

p_m^i is the probability of mutation at i^{th} bit for uniform mutation.

With uniform mutation, for every bit, average change in the value of the individual due to mutation will be equal to each other.

After mutation, one generation of GA is complete. The algorithm starts all over again from the selection process (Figure 2.5). After a predefined number of generations, the algorithms is stopped and the individual with the best fitness till current generation (i.e. $g=G$) is taken as the best optimal result.

$$f_{best}^* = f(x_1^*, x_2^*, ..., x_\eta^*) \tag{2.42}$$

where,

f_{best}^* is the best fitness till current generation,

f is the objective function,

x_i^* is the i^{th} decision variable of the best fitness till current generation.

2.2.2 Simulated Annealing

Simulated annealing is a global optimization method based on the analogy of annealing and cooling of metals. When cooling metals, the molecules move very rapidly while temperature is high and move slowly when cooled down (Van Laarhoven and Aarts, 1987). SA advances point by point and each time comparing the new point with the early accepted point. The point here refers to set of decision variables that corresponds to a unique solution to the problem (i.e $X=(x_1,x_2,.....x_\eta)$; where X is a solution vector). The improvement of solution is achieved by generating random small placements in an iterative fashion (Van Laarhoven and Aarts, 1987). Starting from a random initial point in search space, the value of the objective function is calculated and evaluated for each new displacement. If there is an improvement,

the solution is automatically accepted. If not, it is accepted with a probability. The criterion is called Metropolis Criteria and for a minimization problem, it is given in Eqn. 2.43.

$$p_{acp} = e^{\frac{-(f_{new} - f_{acp})}{T}} \tag{2.43}$$

where,

p_{acp} is the probability of acceptance of the new solution,

f_{new} is the cost of the new solution,

f_{acp} is the cost of the previously accepted solution,

T is a control parameter.

As seen in Eqn. 2.43, control parameter T, which refers to the temperature in the physical process has an influence on the probability of the acceptance of a solution. An initial temperature should be chosen to permit a high rate of transition acceptance, which means a broad covering of the solution space. As the procedure continues, the temperature is lowered and the rate of acceptance will decrease.

The structure of the SA algorithms may vary. However, all of them have a sequence of Metropolis algorithms (Markov chains) that progress at a sequence of decreasing control parameter, T (Van Laarhoven and Aarts, 1987). Basic parameters related to SA are as follows; i) Initial value of control parameter, T ii) Final value of T for convergence, iii) Length of Markov chains and iv) A rule for the change of the control parameter T. The choices related to the above parameters are called as "Cooling Schedule" and greatly affects the performance of the algorithm (Van Laarhoven and Aarts, 1987). General flowchart of a traditional SA algorithm is shown in Figure 2.6.

In traditional gradient based methods, the success of the solution depends on the starting point. The method may stuck in a local optimum according to the initial point chosen. GA intends to surpass this bottleneck by using population based optimization and operators like crossover and mutation. On the other hand, SA, being a point based search method, intends to escape the local minima by not only accepting the better results as in gradient methods, but also the worse ones with some probability. Similar to GA, no requirement for derivatives and easy implementation are the main advantages of SA.

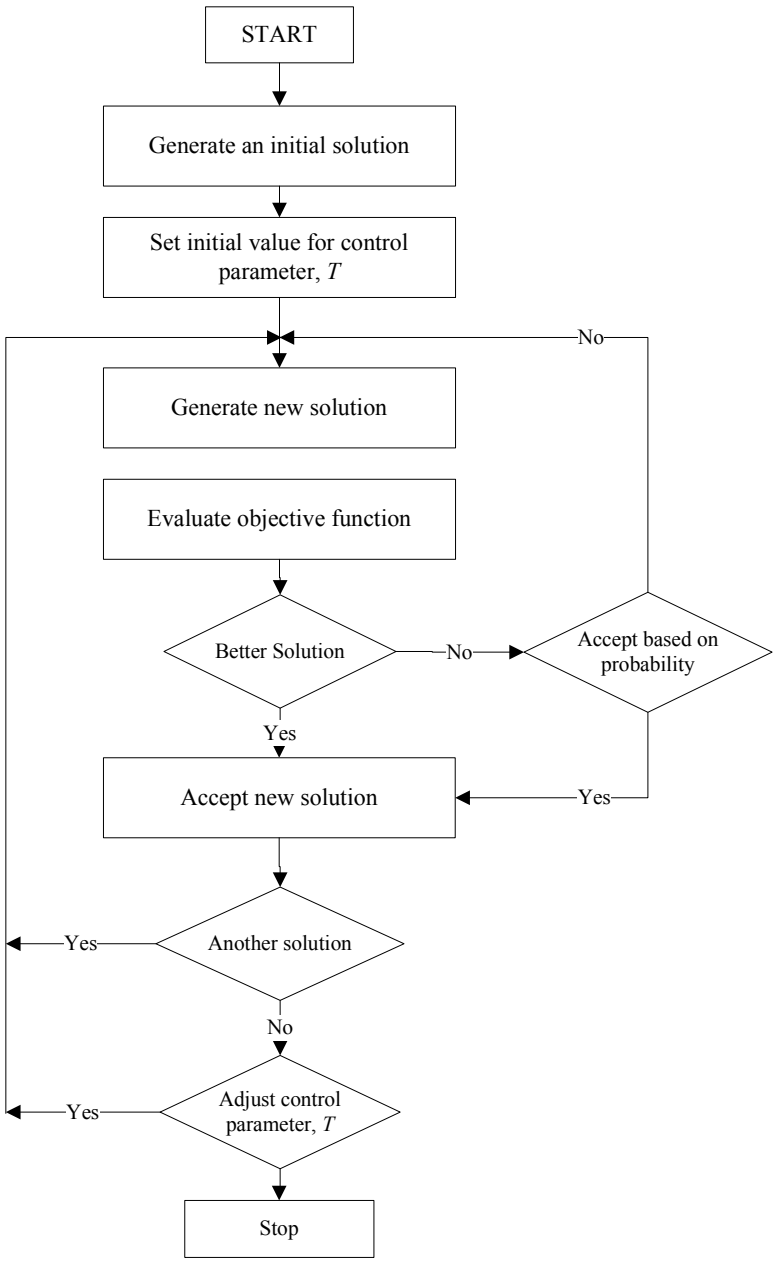


Figure 2.6 General flowchart of SA

Algorithm as derived by Kirkpatrick (1983) was for the solution of a combinatorial problem. Combinatorial problems are defined as the optimization of the arrangement, grouping, ordering and selection of discrete objects usually finite in number (Lawler, 1976). An example to the combinatorial optimization is the famous benchmark problem, traveling salesman. In the problem, minimum traveling distance is searched for a salesman who has to visit a number of cities in a row. For optimization with continuous variables, a set of real values that are usually constrained in a specified interval are utilized. Optimization with continous and combinatorial decision variables are classified in two different optimization groups and the methods for their solution are reported to be quite divergent (Papadimitriou and Steiglitz, 1982).

The first implementations of SA involving continuous decision variables are introduced by Vanderbilt and Louie (1984), Bohachevsky et al. (1986) and Corona et al. (1987). Among these studies, Goffe et al. (1992) defined the implementation of Corona et al. (1987) as the best combination of ease of use and robustness. SA code developed here is based on the algorithm introduced by Corona et al. (1987) and later used by Goffe et al. (1992).

The algorithm starts from a random initial point, X_o. The value of the objective function at the initial point, $f(X_o)$ is evaluated. Initial point is the first accepted point, X_{acp} for SA (i.e $X_{acp}= X_o$ and $f(X_{acp}) = f(X_o)$) The new point, X_{new} is found by making a perturbation to a single decision variable in the solution, in turn (Eqn. 2.44). The remaining variables are left as is. The initial value of the perturbation can be defined by the user according to the range of decision variables.

$$x_i^{new} = x_i + R.v_i \qquad \text{i=1,2}.....\eta \qquad (2.44)$$

where,

x_i^{new} is the new value of the i^{th} decision variable,
R is a random number between 0 and 1,
v_i is step length for i^{th} decision variable.

If the cost of the new point, $f(X_{new})$ is better than the previously accepted solution then the new point is accepted. ($X_{acp}= X_{new}$ and $f(X_{acp}) = f(X_{new})$). If the new point is worse than the previous solution, it is accepted with a probability given by Metropolis criteria (Eqn. 2.43).

As can be seen from Eqn. 2.43, the acceptance probability depends on parameter T and the difference between previous and the new solution. Taking T as constant, if the cost difference between two solutions is high, the probability of acceptance decreases. T is chosen so that early in a run almost all points are accepted. This enables a coarse search of the solution area. Later in a run as T decreases, less number of worse solutions are accepted to direct the solution towards convergence.

After perturbation for each decision variable and corresponding cost functions are evaluated, one whole cycle is completed. The process is repeated for a predefined number of cycles, N_s. Again, cost of each new point is compared with the cost of the previously accepted solution. During this loop, step lengths are constant (howevever, perturbations change according to the random number in Eqn. 2.44). The number of acceptance for each decision variable, κ_i^{accp} is recorded. After N_s cycle, step lengths for each decision variable are updated using the ratio of κ_i^{accp} to the total number of evaluations for each decision variable, which is N_s. If this ratio is greater than 0.6, the step lengths are enlarged and if they are less than 0.4 they are reduced according to the Eqns. 2.45 and 2.46. In between, step lengths remain same.

$$v_i = v_i.(1 + \omega.\frac{\kappa_i^{accp} / N_s - 0.6}{0.4}) \qquad \text{if } \kappa_i^{accp} > 0.6N_s \qquad i=1,2....\eta \qquad (2.45)$$

$$v_i = \frac{v_i}{1 + \omega.\frac{0.4 - \kappa_i^{accp} / N_s}{0.4}} \qquad \text{if } \kappa_i^{accp} < 0.4N_s \qquad i=1,2....\eta \qquad (2.46)$$

where,

ω is a constant parameter usually chosen between 1 and 2.

The new points are found by updating the decision variables one at a time using Eqn. 2.44. The loop is repeated for N_t times. After $N_t \times N_s \times n$ cycles (i.e changing the perturbation N_t times for each decision variable), the value of control parameter T is updated according to the following equation;

$$T = t_{const}.T \qquad (2.47)$$

where,

t_{const} is usually a constant value between 0 and 1 and usually taken close to 1.

Then, with the newly found T and the last perturbations found in previous iterations, it starts all over again. But this time, optimum solution vector till that time, X_{opt} is taken as the initial solution for the new loops ((X_{acp}= X_{opt} and $f(X_{acp}) = f(X_{opt})$). The algorithm can be stopped after a predefined number of T adjustments or after a number of T adjustments where the cost value does not change greater than a percent defined by the user. The flowchart of the algorithm is given in Figure 2.7.

The algorithm as explained above is modified so that T values are updated together with the step-sizes so that the very outer loop has dropped from the algorithm. This way number of function evaluations decreased in a significant amount. The best values found by the regular and modified algorithm code was identical; thus, modified algorithm is used in rest of the study. A code is developed in Fortran 90 programming language based on the above assumptions and algorithm as explained in Section 2.2.2. Annealing parameters are chosen based on suggestions given by Corona et al. (1987), Goffe et al. (1992) and numerous runs with the code. The parameter set used in the study is as follows, N_s=30, N_t=50, ω=2 and initial T value is 1000 and t_{const}=0.85.

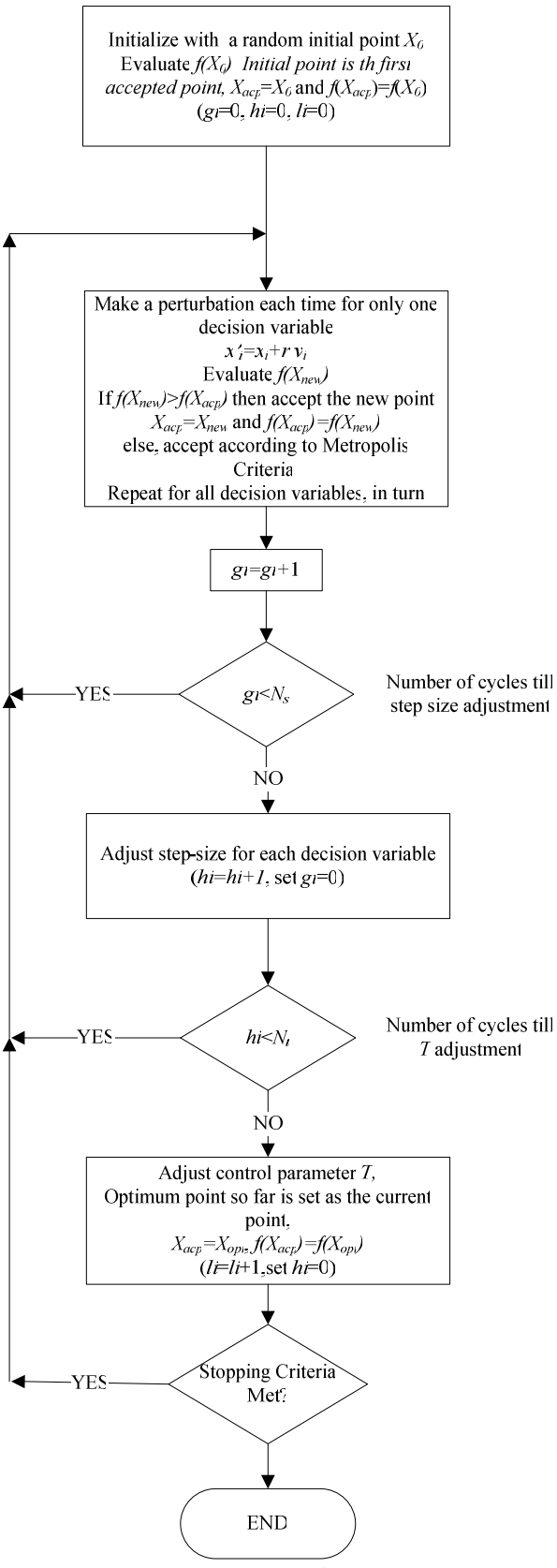


Figure 2.7 Flowchart of continuous SA

CHAPTER 3

GA CODE

3.1 Introduction

The code developed is written in Fortran 90 programming language. The code basically consists of a main program, named “Main” and a subroutine, named “Function”. Main, as indicated by the name, includes the main GA and operators; selection, crossover and mutation. Function includes the commands required to estimate the objective function. The objective function is calculated by calling a simulation model like MODFLOW and running it for each set of solution vectors.

Using the code, one-point, two-point or uniform type of crossover can be performed. Selection types include fitness proportionate selection (with or without scaling) and tournament selection. Two different types of mutation is possible; a classical bitwise mutation with probabilities of mutation for each bit are equal and so called uniform mutation with probabilities for each bit are modified so that the mean change in the value of decision variable remains the same. Elitism can be applied to bring the best individual in current generation to the next generation without visiting selection.

In this study, since computational time is expected to be dominated by the time spent for the calculation of the objective function, the code structure is kept as simple as possible to enable easy modification. Special care is taken to minimize the time spent for calling the numerical model and storing its outputs.

3.1.1 Main Program and Initialization

Main program starts with the statement of constants and variables. Types of different main operators (i.e. type of crossover, mutation or selection, etc.) and additional operators like scaling and elitism are all selected and read through 'input #.txt' file. The values of different GA parameters (i.e. crossover constant, mutation constant, population number, generation number, number of decision variables, string length of each variable, boundary of variables etc.) are also entered through the "input#.txt" file.

The user is free to execute several GA runs with the same input file (i.e. input 1.txt). By this way stochastic effects resulted from random parameters can be tested. In execution of GA, the seed number determines the set of random numbers generated. For the same seed number and input values, GA will find the exact same result. For multiple runs, program automatically updates the seed number, every time a run is complete.

If required, different input files can be built for multiple runs. The files should be created starting from input 1.txt (ie. input 1.txt, input 2.txt .... input n.txt). The number of input files should be stated in the main program.

Initial population is built by randomly assigning 1 and 0 to bits representing an individual. A random number between 0 and 1 which is denoted by R is thrown and if R comes out to be greater than 0.5, the bit takes the value of 1 and else it takes the value of zero. The individuals are stored in two dimensional arrays. First dimension of the array defines the individual's number and the second dimension defines the value of the corresponding bit. The length of string for each variable can be different and set by the user in input file. The procedure is repeated for each member of the population. Initialization code is shown in Figure 3.1.

```
DO i=1,pop_num,1                          ! The procedure is repeated for each member of the
                                          population.
      DO j=1,totstr_length,1
         IF(rand().ge.0.5) then           ! If random number thrown is greater than 0.5, the bit
            birey(i,j)=1.                 takes the value of 1.
         ELSE                             ! If random number thrown is less than 0.5, the bit
            birey(i,j)=0.                 takes the value of 0.
         END IF
       END DO

END DO
```

Figure 3.1 Initialization code

3.1.2 Decoding the Variables and Calculating Fitness

The decision variables are first decoded to the base 10 values (Eqn. 2.25) and then to the real values (Eqn. 2.26). The decoded values are then passed to "Function" subroutine and used in estimating the objective function. The code related to decoding of the variables is shown in Figure 3.2. The objective function is calculated by calling the numerical model; MODFLOW-2000 (Harbaugh et al. 2000). The code waits for the execution of MODFLOW-2000, since the outputs are used in estimating the objective function. A sample of the function subroutine is shown in Figure 3.3.

```
DO k2=1,x_num                             ! "bi-constant" values are calculated at the
      bi_constant(k2)=(up_bound(k2)-      beginning of "Main" program, after the
      low_bound(k2))/(2**str_length(k2)-1) "input.txt" file is read. They are used in
      totstr_length=str_length(k2)+totstr_length calculating the value of decision variables
END DO                                    from their base 10 correspondence.

DO 600 p=1,pop_num,1                      ! The procedure is repeated for each member
                                          of the population.
      top_stra=1
      top_strb=0

         Do xd=1,x_num
            toplam(xd)=0
            top_strb=top_strb+str_length(xd)

               DO j=top_stra,top_strb     !First, integer values of the binary coded
                  toplam(xd)=toplam(xd)+  decision variables are calculated
                  birey(p,j)*2**(top_strb-j)
               END DO

            x(p,xd)=bi_constant(xd)*toplam(xd) !Then, real value of the decision variables
            +low_bound(xd)                are calculated.
            top_stra=top_stra+str_length(xd)
         END DO

600 END DO
```

Figure 3.2 Decoding of the variables

```
result = RUNQQ('c:\modflow\\mf2k.exe','101_inj03.nam')     ! Numerical model is run for each individual
                                                           (i.e. MODFLOW executable file "mf2k.exe"
                                                           with the MODLFLOW list file,
                                                           101_inj03.nam. The "*.nam" file lists the
                                                           packages and input files that is required to
                                                           run MODFLOW). The outputs of the model
                                                           is required for the calculation of violation of
                                                           constraints.

      OPEN (UNIT=8,FILE="101_inj03.hed")                   ! Output file of the numerical model is read
                                                           and required parameters (i.e. head values
            DO I=1,30                                      which represent the potential values in the
            READ(8,*) (HEADS(I,Z),Z=1,60)                  model) are stored in two dimensional
            END DO                                         array(30 is the number of rows while 60 is
                                                           the number of columns in MODFLOW
      CLOSE (UNIT=8,STATUS='KEEP')                         model).

      cp(1)=(8.0078-heads(15,23))                          !Calculation of penalty requires the
      cp(2)=(8.0078-heads(9,29))                           calculation of the degree of the violation for
      cp(3)=(8.0078-heads(21,34))                          constraints (i.e. the difference between the
      cp(4)=(8.0078-heads(6,40))                           potential values at the control points and the
      cp(5)=(8.0078-heads(15,42))                          problem specific potential value of 8.0078,
                                                           which represents toe location))

      g=0                                                  ! Penalty value is initially set to 0.

            DO I=1,5                                       !The penalty for each constraint is summed
            g = g+max(0.0,cp(I))*max(0.0,cp(I))            up. The values are squared to improve the
            END DO                                         efficiency of the optimization process.


      DO I=1,A(1)-3                                        !Objective function is calculated.
            f1=f1-Q(I)
            f2=0
      END DO

      cost = f1 - penalty_coef*g
```

Figure 3.3 Part of the "Function" subroutine that calls "MODFLOW-2000" and calculates objective function

Scaling is used to help selection when the fitness values do not show much variation or to prevent the domination of few individuals when the fitness of these individuals are much higher than the others. Scaling is used for fitness proportionate selection only. Since the order of fitnesses will be unchanged, scaling will have no effect on tournament selection. Linear scaling code is shown in Figure 3.4.

```
fitness_ort=fitness_sum(pop_num)/(pop_num-numfit_zero)   !First, the average fitness of the population is calculated. Since zero fitness individuals are not scaled, they would not be included in scaling calculations.

        fitness_sum(1)=0
        a_constant=(sca_constant-                        !Scaling constant, a is calculated.
1)*fitness_ort/(fitness_eni-fitness_ort)
        b_constant=(1-a_constant)*fitness_ort            !Scaling constant, b is calculated.

        DO p=1,pop_num                                   !Fitness values of all  individuals are scaled.
              fitness(p)=a_constant*fitness(p)+b_constant
              if (fitness(p).le.0) then
              fitness(p)=0
              end if
        END DO

        DO p=1,pop_num-1,1
              fitness_sum(p+1)=fitness_sum(p)+fitness(p+1)
        END DO
```

Figure 3.4 Linear scaling code

3.1.3 Selection

For fitness proportionate selection, first a a random number is thrown between zero and total sum of the fitnesses, f_{sum}. Fittnes values are added one by one until the sum is greater than or equal to the random number chosen. The last individual added is chosen as the first parent that will pass to crossover. The other parent is chosen in the same way. Code for fitness proportionate selection is shown in Figure 3.5.

```
        sel_num1=rand()*fitness_sum(pop_num)     ! A random number is thrown between 0 and total sum of the fitnesses.
        k=1

        DO WHILE(sel_num1.ge.fitness_sum(k))     ! First parent is selected.
          k=k+1
        END DO

        sel_int1=k

        sel_num2=rand()*fitness_sum(pop_num)     ! The above procedure is repeated for the second parent.
        k=1

        DO WHILE(sel_num1.ge.fitness_sum(k))
          k=k+1
        END DO

        sel_int1=k
```

Figure 3.5 Code of fitness proportionate selection

For tournament selection, two individuals are selected randomly. Fitness values are compared and fitter individual passes to crossover (Figure 3.6)

```
sel_num1=rand()*pop_num                          ! Two individuals are  randomly  selected for
sel_num2=rand()*pop_num                          tournament process.
sel_intara1=sel_num1+1
sel_intara2=sel_num2+1


if (fitness(sel_intara1).gt.fitness(sel_intara2)) then     ! Fitter individual is passed to crossover.

     sel_int1=sel_intara1

else

     sel_int1=sel_intara2

end if

sel_num1=rand()*pop_num                          ! The procedure is repeated for the selection of
sel_num2=rand()*pop_num                          second individual
sel_intara1=sel_num1+1
sel_intara2=sel_num2+1

if (fitness(sel_intara1).gt.fitness(sel_intara2)) then

     sel_int2=sel_intara1

else

     sel_int2=sel_intara2

end if
```

Figure 3.6 Code of tournament selection

3.1.4 Crossover

For each pair selected for mating, *R*, a random number between 0 and 1, is thrown. If $R<p_c$, crossover is performed. The point of crossover is set by throwing the dice once again. The bits after the chosen point are interchanged between the mates. If there is no crossover, the mates are left as is (Figure 3.7).

```
r=rand()                                    !First, a random number is thrown between 0
                                            and 1.
                                            !If random number is less than or equal to
    IF (r.le.x_over) THEN                   crossover constant then crossover is
                                            performed.

        rnew=rand()                         !A new number is thrown to estimate the point
        x_overpt=(totstr_length-1)*rnew+1.  of crossover.

        DO c=1,x_overpt,1                   !The bits till the crossover point is kept same.
        Birey_ara(2*p-1,c)=Birey(sel_int1,c)
        Birey_ara(2*p,c)=Birey(sel_int2,c)
        END DO

        DO c=x_overpt+1,totstr_length,1     ! The bits after the point are interchanged
        Birey_ara(2*p-1,c)=Birey(sel_int2,c) between parents.
        Birey_ara(2*p,c)=Birey(sel_int1,c)
        END DO


    ELSE                                    ! If there is no crossover, the parents are left
        x_overpt=0                          without adjustment.
        DO c=1,totstr_length,1
        Birey_ara(2*p-1,c)=Birey(sel_int1,c)
        Birey_ara(2*p,c)=Birey(sel_int2,c)
        END DO

    ENDIF
```

Figure 3.7 The code that performs one-point crossover

For two point crossover, *R* is thrown twice to choose two different points on the strings. The bits between chosen points are interchanged between mates (Figure 3.8).

```
r=rand()                                    ! A random number is thrown between 0 and 1.

IF (r.le.x_over) THEN                       ! If random number is less than or equal to
                                            crossover constant then crossover is performed.
    rnew=rand()                             ! Two different random numbers are thrown to
    rnew2=rand()                            estimate the two points of crossover.

    if (rnew.gt.rnew2) then                 ! Random numbers are put into order.
        rara=rnew2
        rnew2=rnew
        rnew=rara
    end if

    x_overpt=(totstr_length-1)*rnew+1.      ! The points of crossover are set.
    x_overpt2=(totstr_length-1)*rnew2+1.

    DO c=1,x_overpt,1                       ! The bits till the first point of crossover are kept
        Birey_ara(2*p-1,c)=Birey(sel_int1,c) same.
        Birey_ara(2*p,c)=Birey(sel_int2,c)
    END DO

    DO c=x_overpt+1,x_overpt2,1             !The bits from the first to the second point of
        Birey_ara(2*p-1,c)=Birey(sel_int2,c) crossover are interchanged.
        Birey_ara(2*p,c)=Birey(sel_int1,c)
    END DO

    DO c=x_overpt2+1,totstr_length          ! The bits after the second point of crossover
        Birey_ara(2*p-1,c)=Birey(sel_int1,c) are kept same.
        Birey_ara(2*p,c)=Birey(sel_int2,c)
    END DO

ELSE                                        ! If there is no crossover, the parents are left
    x_overpt=0                              without adjustment.
    x_overpt2=0
    rnew=0
    rnew2=0

    DO c=1,totstr_length,1
        Birey_ara(2*p-1,c)=Birey(sel_int1,c)
        Birey_ara(2*p,c)=Birey(sel_int2,c)
    END DO

END IF
```

Figure 3.8 The code that performs two-point crossover

For uniform crossover, a random dummy individual is generated. The string length of this dummy individual is equal to the string length of the individuals in population. For each bit of the mating individuals, corresponding value of the dummy variable is inspected. If the value is 1, the bits are interchanged, otherwise they are left as is. The code that performs uniform crossover is shown in Figure 3.9.

```
r=rand()                                    ! A random number is thrown between 0 and 1.

IF (r.le.x_over) THEN                       ! If the random number is less than or equal to
                                            crossover constant than crossover is performed.

    DO j=1,totstr_length,1                  !Random dummy individual is generated.
        IF(rand().ge.0.5) then
            uni_xover(j)=1.
        ELSE
            uni_xover(j)=0.
        END IF
    END DO

    DO c=1,totstr_length,1                  !For each bit, whether there will be crossover or
                                            not is inspected.

        if (uni_xover(c).eq.1) then !make xover   !If the bit value of dummy individual is 1, the
            Birey_ara(2*p-1,c)=Birey(sel_int2,c)  corresponding bits in mating individuals are
            Birey_ara(2*p,c)=Birey(sel_int1,c)    interchanged.

        else if (uni_xover(c).eq.0) then    !Else, they are left as is.
            Birey_ara(2*p-1,c)=Birey(sel_int1,c)
            Birey_ara(2*p,c)=Birey(sel_int2,c)
        end if

    END DO

ELSE                                        ! If there is no crossover, the parents are left
    x_overpt=0                              without adjustment.
    DO c=1,totstr_length,1
        Birey_ara(2*p-1,c)=Birey(sel_int1,c)
        Birey_ara(2*p,c)=Birey(sel_int2,c)
    END DO

ENDIF
```

Figure 3.9 The code that performs uniform crossover.

3.1.5 Mutation

By moving bit by bit through strings, it is decided whether there will be a mutation or not. For each bit, R is thrown and if $R<p_m$, the bit is changed (i.e. if it is one, it becomes 0, else it becomes 1). Otherwise, it remains unchanged (Figure 3.10).

```
DO 400 ix=1,pop_num,1                    ! Mutation procedure is repeated for each individual.
        mut_adet=0                       ! Number of mutations occurred are stored for inspection.
        DO jx=1,totstr_length,1

         rastgele=rand()                 ! If random number is less than or equal to mutation number, then mutation is performed.

         if(rastgele.le.mut_num) then
               if (birey(ix,jx).EQ.1) then   ! The bits are flipped.
                 birey_ara(ix,jx)=0
                 mut_adet=mut_adet+1
               else
                 birey_ara(ix,jx)=1
                 mut_adet=mut_adet+1
               endif
         END IF
        END DO
  400   END DO
```

Figure 3.10 The code that performs mutation

In uniform mutation, the probability of mutation in a bit depends on its location. The individuals are mutated according to Eqn. 2.41. Code of uniform mutation is shown in Figure 3.11.

```
DO 700 ix=1,pop_num,1                    ! Mutation procedure is repeated for each individual.
        mut_adet=0                       ! Number of mutations occured is stored for inspection.
        top_stra=1
        top_strb=0
    DO xd=1,x_num
        top_strb=top_strb+str_length(xd)
        DO jx=top_stra,top_strb

            rastgele=rand()
            IF (rastgele.le.mut_num/(2**(top_strb-jx+1))) THEN   ! If random number is less than or equal to modified mutation number, then mutation is performed.

                IF (birey(ix,jx).EQ.1) THEN   ! The bits are flipped.
                    birey_ara(ix,jx)=0
                mut_adet=mut_adet+1
                ELSE
                    birey_ara(ix,jx)=1
                    mut_adet=mut_adet+1
                END IF
            END IF
        END DO
    top_stra=top_stra+str_length(xd)
    END DO
700 END DO
```

Figure 3.11 The code that performs uniform mutation

3.1.6 Output Files

To store the data related to the current run and to inspect the code's efficiency, different output files are created. The files are stored in a subdirectory of "sonuclar" directory, which is in the same directory that the program is executed. Subdirectory for each run is separate and automatically created.

The main output file is "output.txt". The binary-coded individuals, their fitnesses, penalty values, best fitness and average fitness values are listed generation by generation in "output.txt" file. Average and best fitness values are also printed explicitly to "best_fitness.txt" and "average_fitness.txt" files. Crossover and mutation processes are printed in "xover_control.txt" and "mutas_control.txt" files, respectively. Since these files can be quiet large for excessive number of populations, generations or string numbers as well, it is recommended to comment out the related parts in the code, unless detailed examining is required. Additional output files are created if multiple runs are executed to enable easy comparison (i.e. avg_all.txt and best_all.txt). All the files are prepared in a format that data can be easily transferred to common spreadsheets.

The program can easily be adopted to solve different optimization problems. Initially, the inputs should be arranged for the number of decision variables (i.e. string length, boundary values, etc.). Then, the old objective function should also be replaced with the new function. If an external solver is to be included, associated codes should be rewritten for calling the new solver and utilizing the related outputs for objective function estimation.

CHAPTER 4

OPTIMIZING GA PARAMETERS

4.1 Dewatering Example

To scrutinize the developed code and optimize the GA parameters for the rest of the study, a simple circular idealized aquifer with radius 1000 m is utilized (Figure 4.1). The problem is a hypothetical problem, which was considered by Demirbas (2003). The objective is to estimate the total extraction rates for the wells in order to lower the water table to specified values at control points. Circular boundary of the aquifer has constant piezometric head that is 20 m. There are three wells; W_{EXT1}, W_{EXT2}, W_{EXT3} and two control locations; CP_1 *and* CP_2 (Table 4.1). W_{EXTi} denotes the i^{th} extraction well and CP_j denotes the j^{th} control point. Control locations represent the points where hydraulic head is of importance. Hydraulic conductivity of the aquifer is constant and its value is 11.448 m/day.

Table 4.1 Locations of the wells and control points

Well and control points	*x* (m)	*y* (m)
W_{EXT1}	950	1050
W_{EXT2}	1350	1050
W_{EXT3}	1050	1550
CP_1	1050	1150
CP_2	1050	1350

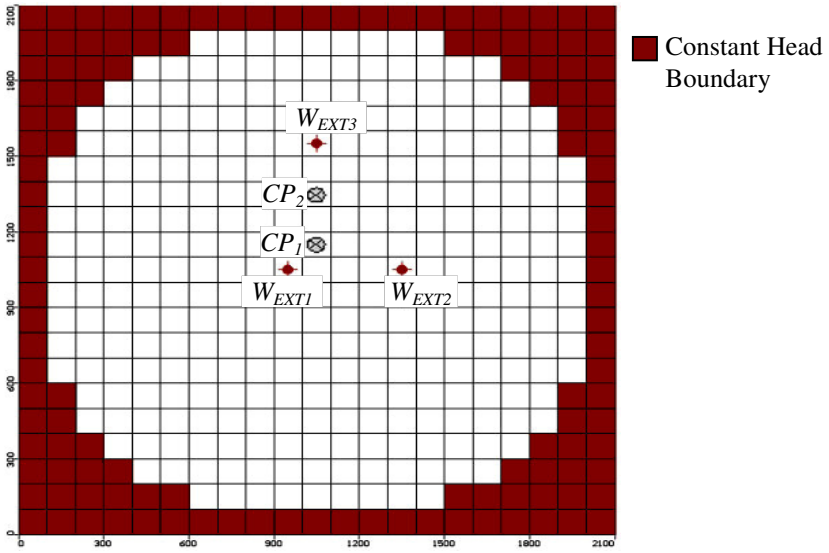


Figure 4.1 Plan view of the aquifer

The water table is required to be lowered to at least h_1^u at CP_1 and to at least h_2^u at CP_2.. Maximum and minimum capacities of the wells are limited to Q_{EXTi}^u and Q_{EXTi}^l, respectively. The objective is to minimize the total discharge rate for above conditions.

The formulation for optimization problem is as follows;

The objective function,

$$Minimize\ f = \sum_{i=1}^{3} Q_{EXTi} \tag{4.1}$$

Set of constraints,

$$Q_{EXTi}^l \le Q_{EXTi} \le Q_{EXTi}^u \qquad \text{for } i=1,2,3 \tag{4.2}$$

$$h_{CPj} \le h_j^u \qquad \text{for } j=1,2 \tag{4.3}$$

where, h_{CPj} is the head at j^{th} control point. h_1^u and h_2^u values are 18 m and 18.5 m, Q_{EXTi}^u and Q_{EXTi}^l values are 0 and 1500 m^3/day, respectively.

The management problem formulated above is then converted into an unconstrained problem. Since constraints on discharge values (Eqn. 4.2) are defined as boundaries of

decision variables, only constraints on head values (Eqn. 4.3) are added to the objective function as penalty function.

Unconstrained objective function,

$$Minimize\ f = \sum_{i=1}^{3} Q_{EXTi} + c\sum_{j=1}^{2} (\max(0, h_{CPj} - h_j^u))^2 \tag{4.4}$$

"c" is a problem dependent constant and choosing the right penalty constant requires much attention. High penalty constants result in distortion in solution space (Goldberg,1989). As the penalty constant increases, the unconstrained formulation approaches the constrained solution where infeasible solutions are omitted from the optimization process. Doing so throws away the information that can be provided from the infeasible solutions. On the other hand, very small penalty constants may result in infeasible solutions since infeasible solutions with lower penalties may have better fitness with respect to the feasible solutions. This trade-off increases the difficulty of deciding on penalty constant. The penalty constant in this study is obtained by increasing it till no infeasible optimal solution is achieved (Table 4.2). At the end it is expected that best feasible string will be more fit than an infeasible string with low constraint violation.

Table 4.2 The characteristic of GA for different penalty values

Penalty constant	Number of infeasible optimals among 20 runs
10,000	20
50,000	20
100,000	13
150,000	0
250,000	0

GA code developed can only solve maximization problems. Eqn. 4.4 can be transformed to a maximization problem by multiplying the right hand side of equation with minus one. However, fitness values should be positive in order that GA operates correctly (i.e. for proper ordering of fitness values in fitness proportionate selection). Thus, a positive large number is added to the right hand side of equation. This number should be larger than the sum of upper values of decision variables (i.e. $\sum_{i=1}^{3} Q_{EXTi}^u$). Thus, solutions with no or little

constraint violation are not omitted from the selection process. This number should not to be selected very large, so that selection process can easily make a choice between good and better individuals. Here, the number is taken as 10000, although a wide range of selection can be made according to the above-mentioned criteria. Eqn. 4.4 then takes the form;

$$Maximize\ f = 10000 - \sum_{i=1}^{3} Q_{EXTi} - c\sum_{j=1}^{2} (\max(0, h_{CPj} - h_j^u))^2 \tag{4.5}$$

"Function" subroutine in GA code is modified according to Eqn. 4.5.

A particular run of optimization problem is considered to show the step by step progress of GA. To start off, a randomly generated initial population of 16 individuals is shown in Table 4.3.

Table 4.3 An example of an initial population generated by GA

Indv. no.	Individuals
1	1 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 1
2	1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1
3	0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0
4	1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0
5	0 0 1 1 0 0 0 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 1
6	0 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 1 0
7	1 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 1 1
8	1 0 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0
9	1 1 1 1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0
10	1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 0 0 1 1 1 1
11	1 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1
12	0 0 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1
13	1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1
14	0 0 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 0 0 1 0
15	0 0 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 1 0
16	0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1

As an example, binary coding of the 4th individual is

$v_4 =$ 1 1 1 0 0 0 0 1 | 0 0 1 0 1 0 0 1 | 1 0 1 1 1 1 1 0

For this problem, number of bits representing each variable is constant and set to eight. The first eight bits represent the first decision variable. Next eight bits represent the second and remaining bits represent the third decision variable. The number of bits representing each variable is set at the beginning of a GA run.

z_1 = 1 1 1 0 0 0 0 1
z_2 = 0 0 1 0 1 0 0 1
z_3 = 1 0 1 1 1 1 1 0

The base 10 value of z_1 is calculated by using the Eqn. 2.25.

$(z_1)_2 = (r_1)_{10}$
$(1\ 1\ 1\ 0\ 0\ 0\ 0\ 1)_2 = (1x2^0+0x2^1+0x2^2+0x2^3+0x2^4+1x2^5+1x2^6+1x2^7)_{10}$
$r_1 = 225$

x_1 is then calculated using Eqn. 2.26.

$$x_1 = 0 + 225.\frac{(1500-0)}{2^8-1} = 1323.53 \text{ m}^3\text{/day}.$$

(1500-0) is the range of variable x_1 and (2^8-1) is the number of intervals that this range will be divided into. The value of $(1500-0)/(2^8-1) = 5.88$ m^3/day then gives the precision of x_1. In other words for potential solutions, x_1 can take the value of 0 or 5.88 m^3/day, but no other value in between. The precisions for the remaing variables, x_2 and x_3 are also the same, since the string lengths and the ranges for the variables are the same. To increase the precision, either string length should be increased or range of the variables should be narrowed (Coley, 1999). Similarly, x_2 and x_3 are encoded to their values as 241.2 m^3/day and 1117.7 m^3/day, respectively. Base 10 and real values of remaining variables are listed in Table 4.4.

Table 4.4 Decoded values of decision variables

Indv. No	z_1	z_2	z_3	r_1	r_2	r_3	x_1 (m^3/day)	x_2 (m^3/day)	x_3 (m^3/day)
1	10101100	01010000	11111011	172	80	251	1011.8	470.6	1476.5
2	10111001	00010001	11011001	185	17	217	1088.2	100.0	1276.5
3	01000011	01110110	10010000	67	118	144	394.1	694.1	847.1
4	11100001	00101001	10111110	225	41	190	1323.5	241.2	1117.6
5	00110000	11001011	01011111	48	203	95	282.4	1194.1	558.8
6	00010101	01001000	00101010	21	72	42	123.5	423.5	247.1
7	11110001	00101100	11111011	241	44	251	1417.6	258.8	1476.5
8	10101110	00101001	11101010	174	41	234	1023.5	241.2	1376.5
9	11110010	10100010	00000010	242	162	2	1423.5	952.9	11.8
10	11011101	01100010	00001111	221	98	15	1300.0	576.5	88.2
11	10000000	01110101	10010111	128	117	151	752.9	688.2	888.2
12	00110111	01000101	10101001	55	69	169	323.5	405.9	994.1
13	10110001	11010000	10011001	177	208	153	1041.2	1223.5	900.0
14	00010111	01011111	01110010	23	95	114	135.3	558.8	670.6
15	00100110	01111100	01000010	38	124	66	223.5	729.4	388.2
16	00111111	11110111	10110101	63	247	181	370.6	1452.9	1064.7

After decoding the decision variables for a solution (i.e. discharge values for extraction wells), objective function is evaluated using Eqn. 4.4. MODFLOW run is required to estimate the head values at control points. Head values are used to calculate the penalty for constraint violations (Eqn. 4.3). The objective function for the 4th individual is calculated as,

$f_1 = f(1323.5\ m^3/day,\ 241.2\ m^3/day,\ 1117.7\ m^3/day) = 7317.7$

Total discharge rates, fitnesses and penalty values calculated for the initial population are shown in Table 4.5. From the table, it can be seen that penalty is applied to individual 3, 5, 6, 12, 14 and 15. For these solutions, total water extracted is not high enough to lower the heads to required levels. Fitness of individuals 6 and 14 come out to be negative due to high degree of violation and these individuals are omitted from selection process. However, individual 3, 5, 12 and 15 would still have chance to be selected.

Table 4.5 Fitness and penalty values for initial population

Indv. No.	Individuals	Fitness	Total Q (m^3/day)	Penalty value
1	1 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 1	7041.2	2958.8	0
2	1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1	7535.3	2464.7	0
3	0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0	7280.7	1935.3	800
4	1 1 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0	7317.7	2682.4	0
5	0 0 1 1 0 0 0 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 1	7564.7	2035.3	400
6	0 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 1 0	-15351.1	794.1	24600
7	1 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 1 1	6847.1	3152.9	0
8	1 0 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0	7358.8	2641.2	0
9	1 1 1 1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0	7611.8	2388.2	0
10	1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 0 0 1 1 1 1	8035.3	1964.7	0
11	1 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1	7670.6	2329.4	0
12	0 0 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1	5360.5	1723.5	2900
13	1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 0 0 1	6835.3	3164.7	0
14	0 0 0 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 0	-364.7	1364.7	9000
15	0 0 1 0 0 1 1 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 0	333.8	1341.2	8300
16	0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1	7111.8	2888.2	0

The probabilities of selection for the initial population are given in Table 4.6. The individual with best fitness is v_{10}. v_{10} has a fitness of 8035.3. Zero penalty indicates no violation of the constraints. v_{10} has the highest chance to be selected for mating with a probability of 8.6% (Eqn. 2.28). However, the results show that it is not selected. This is due to stochastic effects

and can be bypassed by choosing elitism. The results also show that probability of selection for most of the individuals (13 individuals among 14 individuals that will be selected for the next generation) are almost identical, ranging from 5.7% to 8.6%. Considering the stochastic effects, selection of any of individual among the population is very close to random selection. Scaling the fitness values, according to average fitness and thus increasing probability interval (range) between good and better individuals (or bad and worse solutions) may enhance the algorithm performance.

Table 4.6 Expected probabilities of selection in initial generation

Indv. No.	Fitness	Probability of selection (%)
1	7041.2	7.5%
2	7535.3	8.0%
3	7280.7	7.8%
4	7317.7	7.8%
5	7564.7	8.1%
6	0.0	0.0%
7	6847.1	7.3%
8	7358.8	7.8%
9	7611.8	8.1%
10	8035.3	8.6%
11	7670.6	8.2%
12	5360.5	5.7%
13	6835.3	7.3%
14	0.0	0.0%
15	333.8	0.4%
16	7111.8	7.6%
Total	93904.4	100.0%

Mating Pairs (Parents)

Crossover rate of 0.85 indicates that about 85% of all individuals are expected to undergo crossover. Initially, individual 2 and 8 are selected for mating (Table 4.7). Since *R* thrown for the pair (0.466) is less than p_c (0.85), it is decided to perform crossover. The point of crossover is decided by another *R* which comes out to be 0.256. This number times length of string in base 10 value gives the point of crossover (i.e. 0.256×24=6.15 and $6.15=(6)_{10}$). In Table 4.7, the point of crossover is shown with a slash sign. The bits after this point are interchanged between mating individuals.

Table 4.7 An example of crossover from the sample problem

	Indiv. No.	Individuals	Fitness Value	Penalty Value
Before Crossover:	8	1 0 1 0 1 1 /**1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0**	7358.82	0
	2	1 0 1 1 1 0 /0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1	7535.29	0
After Crossover:	*A*	1 0 1 0 1 1 /0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1	7605.88	0
	B	1 0 1 1 1 0 /**1 0 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0**	7288.24	0

One of the newly built individuals (Individual *A*) has a better fitness than both of its parents. However, as seen in Table 4.7, this is not always the case (Individual *B*). The idea of crossover is to improve the population by exchanging ideas. However, exchanging of knowledge together with selection only guarantees the population to evolve to better solutions (Coley, 1999).

Mutation

The springs now undergo mutation. The rate of mutation, p_m is 0.004. This indicates that only 1 mutation among 250 bits is expected. Mutation constant multiplied with total string length ($p_m \times L$) gives the expected probability of mutation for an individual which is $0.004 \times 24 \cong 0.1$ (i.e. approximately one among ten individuals is expected to undergo mutation). One of the mutated individuals is shown in Table 4.8. Only one bit (i.e. 10^{th} bit) undergoes mutation, for which the random number thrown, 1.0838988×10^{-3} is less than p_m, 0.004.

Table 4.8 An example of mutation from the sample problem

	Individuals	**Fitness Value**	**Penalty Value**
Before Mutation:	1 1 0 1 1 1 0 1 0 **1** 1 0 0 0 1 0 0 0 0 0 1 1 1 1	8035.29	0
After Mutation:	1 1 0 1 1 1 0 1 0 **0** 1 0 0 0 1 0 0 0 0 0 1 1 1 1	8155.76	0

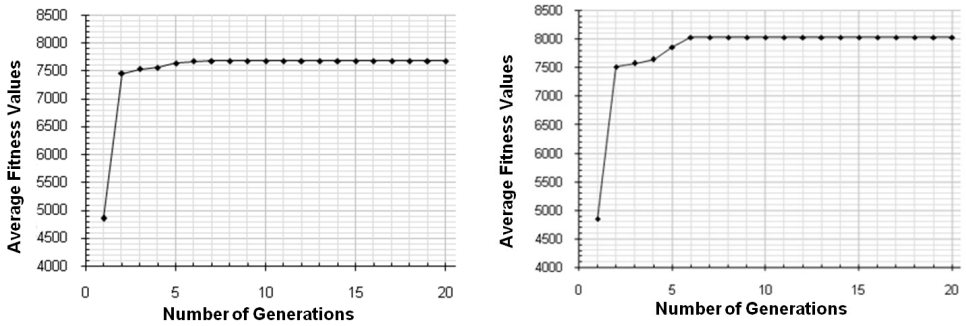
Through Generations

Referring to Table 4.9, it is seen that, at the end of ten iterations, best fitness value increased from 8035.3 to 8215.7. Since, there is no elitism, best fitness of current generation is not monotonically increasing. Average fitness of the population increased from 5869.0 to 7247.9, in one generation. From this generation on, average fitness shows a noisy progress. Note that crossover rate, p_c=0.85 and mutation rate, p_m=0.04 for this case.

Table 4.9 Average and best fitness values through generations

Generation No.	Best Fitness Values	Best Fitness of Current Generation	Average Fitness Values
1	8035.3	8035.3	5869.0
2	8155.8	8155.8	7247.9
3	8155.8	8011.8	7169.7
4	8155.8	7829.4	7112.3
5	8155.8	7929.4	7126.5
6	8155.8	8051.5	6896.2
7	8215.7	8215.7	6901.7
8	8215.7	7900.0	7259.6
9	8215.7	7864.7	6981.4
10	8215.7	7864.7	7187.1

If there were no crossover and mutation, but only selection (i.e. p_c and p_m are set to zero), average fitness of the population would converge to the fitness of a single individual in initial population (Figure 4.2-a). For this case, the population converges to individual 11 (υ_{11}), with a fitness value of 7670.6 (Initial population is the same as the previous case, since seed number used for random number generation is still the same). Since there is no elitism, converged individual needs not to be the best one. However, if elitism is applied, average fitness value converges to the fitness of the best individual; individual 10 (υ_{10}) with a fitness value of 8035.3 (Figure 4.2-b).



a) b)

Figure 4.2 Average fitness values for p_c=0 and p_m=0, a) No elitism b) Elitism is applied

Since random processes are involved in GA (i.e. random numbers thrown to decide in crossover, mutation and selection, initial population generated, etc.) no two runs of GA are same, unless the seed numbers that decide on the random numbers are the same. Thus, single

run is not to be trusted to derive solid conclusions for GA and its parameters. Multiple runs are required to reduce stochastic variations (Coley, 1999). In Figure 4.3, GA results for the sample problem are shown for five different runs. Best optimal results range from 7964.41 to 8281.65.

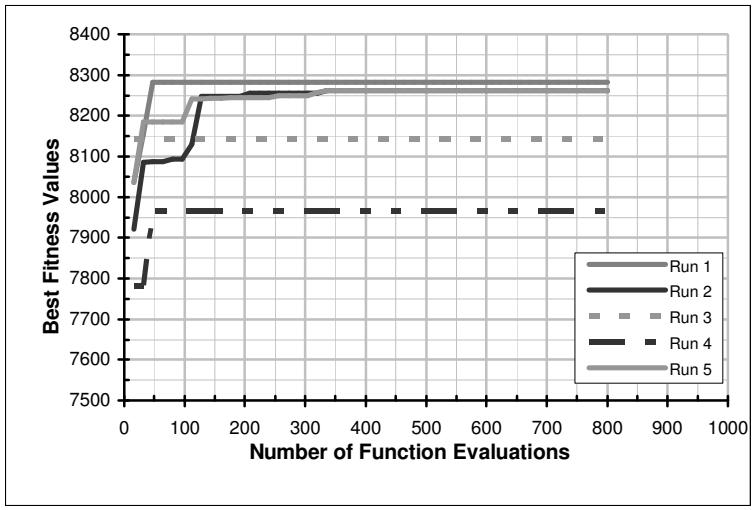


Figure 4.3 GA result from five different runs

4.2 GA Parameters

The wide spectrum of parameter values in literature review (see Section 1.2.3) shows that GA results not only depend on GA parameters but also the problem to be optimized (the characteristic of the search space) or the type of GA operators selected (i.e. their effect on selection pressure). Following the early studies listed in literature review, initial parameter set for starting the GA run is chosen as P=30, p_c=0.6 and p_m=0.0417. The results are averaged over 20 runs.

4.2.1 Effect of Scaling and Elitism

The effect of applying linear scaling and elitism on the dewatering example is shown in Figure 4.4. Not only both methods provided a faster convergence rate and improved solution but using them together gave the best results. Using elitism without scaling only performed better than applying scaling only.

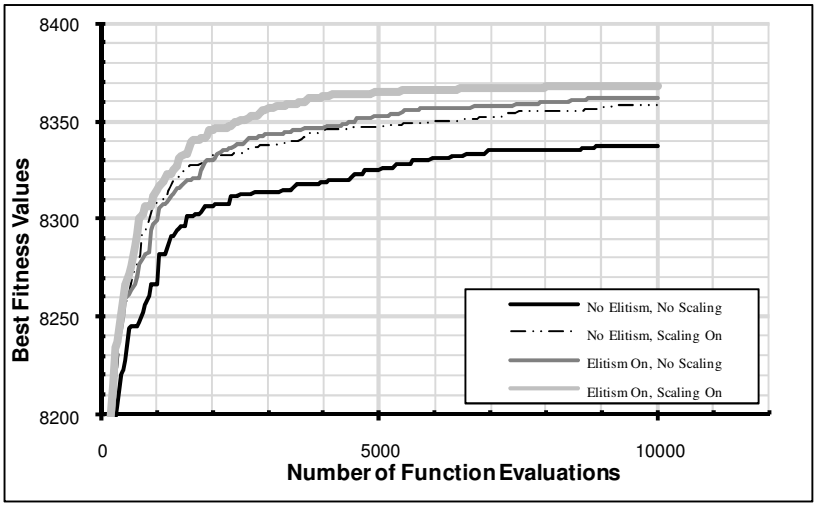


Figure 4.4 Effect of scaling and elitism on optimization (averaged over 20 runs)

4.2.2 Effect of Population Number

Five different population numbers are tested and best fitness rates are shown in Figure 4.5. It is seen that most of the population numbers converge to the same fitness value, 8370.59. Among five different instances, only P=30 does not converge to this value. Set of decision variables for the best solution is x_1=1211.77 m^3/day, x_2=0 and x_3=417.65 m^3/day. For P=50, the algorithm required the least number of function evaluations for convergence.

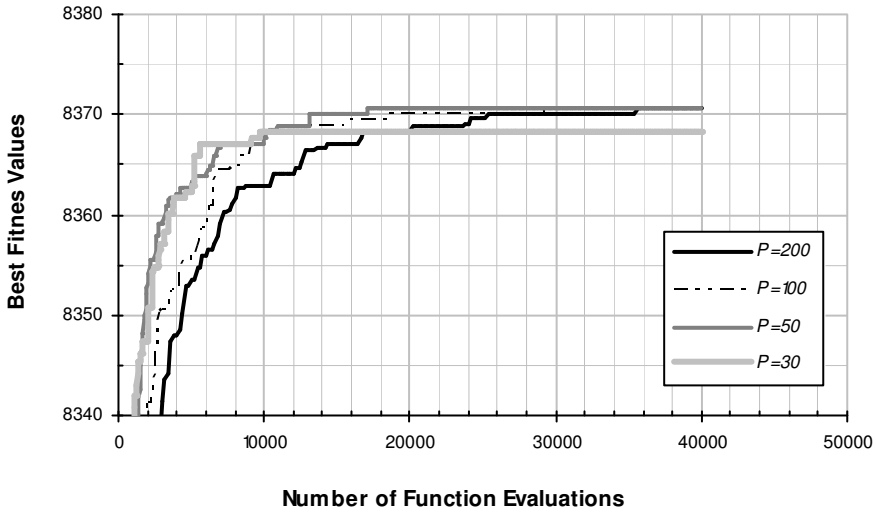


Figure 4.5 Effect of population number on optimization (averaged over 20 runs)

4.2.3 Effect of Crossover Number

For the crossover rate, p_c=0 (no crossover), 0.25, 0.5, 0.75 and 1.0 (mates always undergoes crossover) values were tested. As a result of the previous population test, population number is taken as 50. Referring to Figure 4.6, it is seen that only p_c=0.5 and p_c=0.25 converges to best fitness. p_c=0.5 converges to the best optimal solution faster than p_c=0.25. It is interesting to observe that with no crossover but mutation and selection only, the algorithm converges to an acceptable solution for this sample problem.

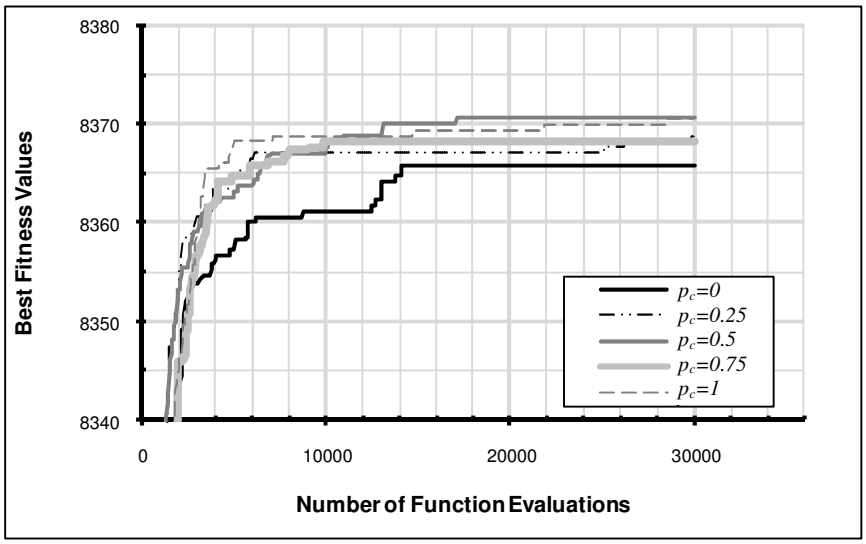


Figure 4.6 Effect of different crossover rate on optimization (averaged over 20 runs)

4.2.4 Effect of Mutation Number

Figure 4.7 shows the effect of different mutation rates on solution. Five different mutation rates including 0.0417 are tried. This value corresponds to 1/total number of bits for the problem. Although all the results are acceptable, the best results are achieved with p_m=0.0417. The rate is similar to what Back and Schutz (1993) suggested.

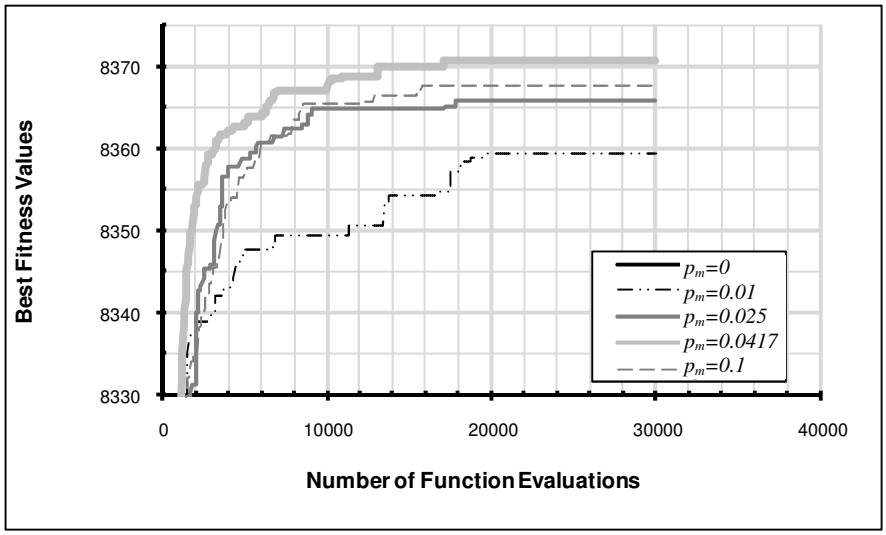
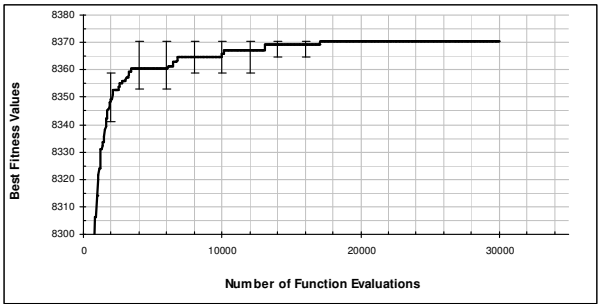


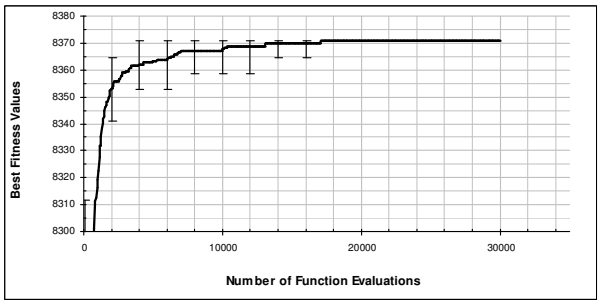
Figure 4.7 Effect of mutation rates on optimization (averaged over 20 runs)

4.2.5 Effect of Number of Runs on Solution

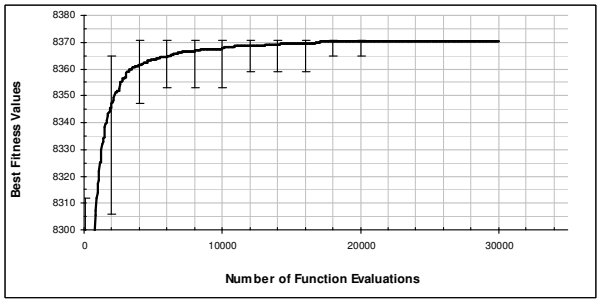
The results for different number of runs are shown in Figure 4.8. For this example, using the optimal combination of parameter set, GA can find the maximum fitness, in every run. Stochastic errors have no effect on finding the optimal solution. However, as the search space gets complicated, the results from different runs are expected to be different. To derive solid conclusions, GA runs should be repeated with different seed numbers (See Section 5.1). Though the optimal solutions found at the end are not different, it can be seen that as the number of runs increased, the figure gets a smother shape, giving a broader idea of the progress of GA for different runs. The average change of best fitness curve from 5 runs to 10 runs is less than 0.02 % while from 10 runs to 15 runs, it is less than 0.009% and from 15 to 20 runs less than 0.007%.



a)



b)



c)

Figure 4.8 The results for different number of runs a) Five runs b) Ten runs c) Twenty runs

Optimal Parameter Set Used for the Rest of the Study

In our studies, it is seen that various combination of parameter sets can be used in order to get adequate results. The following set is accepted as the optimal set and will be primarily used in the problems in following chapters (i.e. fittness proportionate selection, two point crossover, scaling and elitism on, scaling constant, λ_{scal} =1.5, crossover rate, p_c=0.5, mutation rate, p_m=0.0417 (1/Length of string), population number, P=50)

CHAPTER 5

APPLICATION AND DISCUSSION OF RESULTS

A coastal aquifer, previously studied by Mantoglou (2003) is used as the basic model for the optimization of the management models introduced in this thesis. First, the very same problem by Mantoglou (2003) which seeks for the optimal amount of extraction from five different extraction wells is studied. Different seawater prevention strategies (injection wells, canals and both) are added to the model in order to optimize the additional decision variables related to the prevention methods (i.e. injection rate and location for injection wells, recharge rate and location for canals, etc.). The optimization results by GA and SA are compared with the results from LP or MIP, whenever applicable.

5.1 Extraction Wells Only Problem

The example involves the optimal management of extraction wells at specified locations. The simulation model used is a rectangular coastal aquifer in Greek island of Kalymnos. All required parameters and assumptions are taken from the study by Mantoglou (2003). The finite difference representation of the model is shown in Figure 5.1. The sea shore is located on the west. The aquifer which is 7000 m length and 3000 m wide is discretized with constant Δx=116.67 m in East-West direction and constant Δy=100 m in South-East direction (Δx is the distance between two discretization lines in x-coordinate direction and Δy is the distance between two discretization lines in y-coordinate direction). Depth of aquifer is 25 m. The North and South boundaries are taken as impervious. The recharge rates at higher elevations are 150 mm/year over an area of 9 km^2. This recharge is fixed flux boundary on the east side (i.e. 150 mm / year×9 km^2 = 1.35×10^6 m^3/year). There is an additional recharge of 50 mm/year over the aquifer. Hydraulic conductivity is constant as 100 m/day. There are five well locations for which the coordinates are listed in Table 5.1.

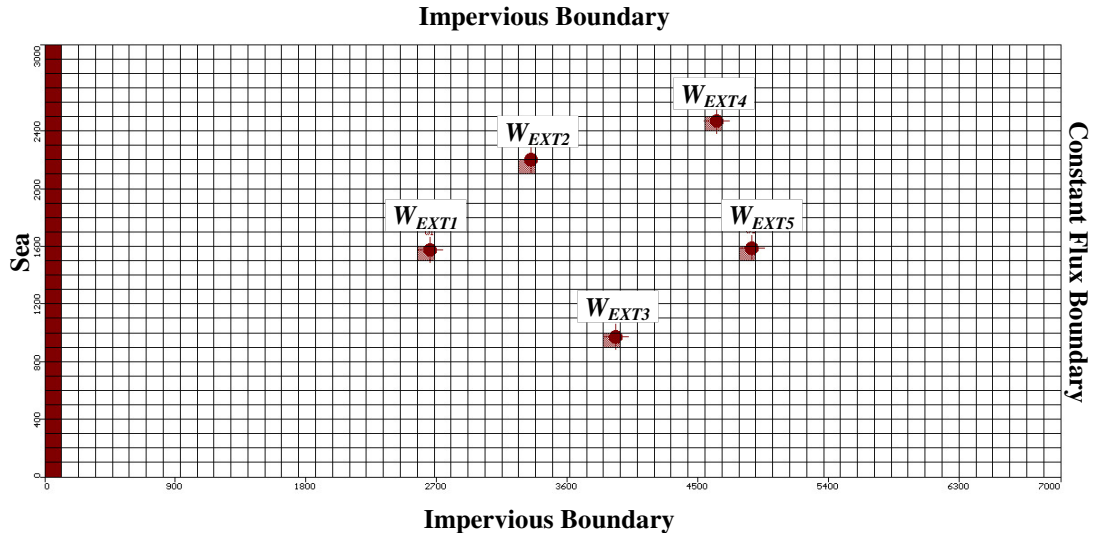


Figure 5.1 Plan view of the aquifer

Table 5.1 Location of the extraction wells

Extraction Wells	*x* (m)	*y* (m)
W_{EXT1}	2657	1572
W_{EXT2}	3353	2200
W_{EXT3}	3932	975
W_{EXT4}	4632	2470
W_{EXT5}	4873	1586

The objective is to find the total maximum rates for the extraction wells without letting the seawater intrusion. ϕ_{toe} represents the border where the seawater intrusion reaches. If the ϕ values at the wells are smaller than ϕ_{toe}, it is assumed that wells are intruded by seawater.

The formulation for the optimization problem is as follows;
The objective function,

$$Maximize\ f = \sum_{i=1}^{5} Q_{EXTi} \tag{5.1}$$

Set of constraints,

$$\phi_i \geq \phi_{toe} \qquad i=1,2\ldots.5 \tag{5.2}$$

$$0 \leq Q_{EXTi} \leq 1500 \text{ m}^3/\text{day} \qquad i=1,2\ldots.5 \tag{5.3}$$

where, ϕ_{toe} is calculated as 8.0078 m^2 for ρ_s =1025 kg/m^3 and ρ_f =1000 kg/m^3 and d=25 m (Eqn. 2.9).

ϕ values are found by using MODFLOW. The boundary conditions are modified according to Eqns. 2.6 and 2.7. At the coast, since h_f=d, ϕ=0. At no flow boundaries $\partial h_f / \partial x$ and $\partial h_f / \partial y = 0$ then $\partial \phi / \partial x$ and $\partial \phi / \partial x = 0$.

The management problem above then converted into an unconstrained problem. Since the upper and lower bounds of the well rates are entered as input to the genetic code, only constraints related to potential values at well locations (state variables) are added to the objective function as penalty term, to obtain the unconstrained objective function,

$$Maximize\ f = \sum_{i=1}^{5} Q_{EXTi} - c\sum_{i=1}^{5} (\max(0, 8.0078 - \phi_i))^2 \tag{5.4}$$

To find the value of the penalty constant, c, different values are tested by making various runs and as explained in Section 4.1, the minimum one that gives no infeasible solution is used as the penalty constant for the rest of the cases studied. Following the above procedure, c is found to be 10000.

5.1.1 Discussion of Results for Extraction Wells Only Problem

20 independent runs are carried out using GA. GA parameters are taken from the optimal results found in Chapter 4 (i.e. P=50, p_m=0.025 (which is 1/L=1/40), p_c=0.5, scaling and elitism are on and two point crossover is applied). The convergence behavior of the best solution found among 20 runs is shown in Figure 5.2.

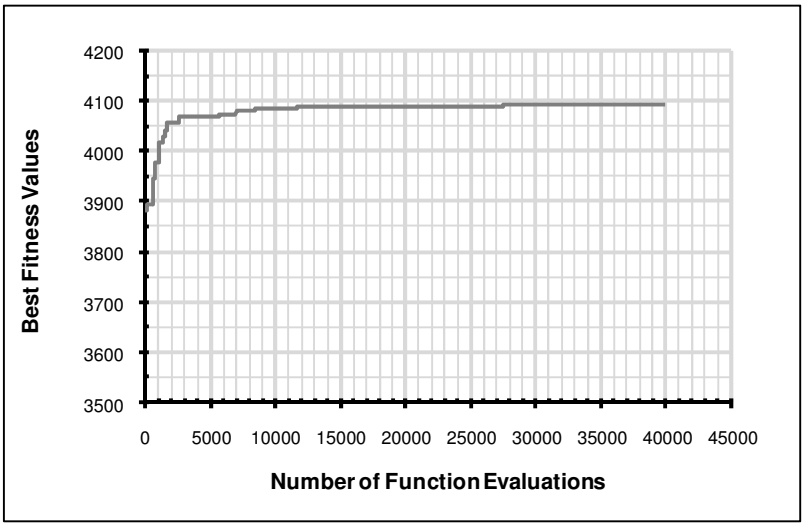


Figure 5.2 Best solution found by GA for Mantoglou (2003) problem

The optimal set of decision variables found by GA is Q_{EXT1}=58.82 m^3/day, Q_{EXT2}=323.53 m^3/day, Q_{EXT3}=882.35 m^3/day, and Q_{EXT4}=1347.06 m^3/day and Q_{EXT5}=1476.47 m^3/day. ϕ values corresponding to the optimal solution is shown in Figure 5.3. ϕ=8.0078 m^2 line which represents the toe of saltwater encroachment can be seen just in front of the extraction well. Using the transformation formulations (Eqns. 2.10 and 2.11), freshwater head values and interface elevations are found and freshwater heads above seawater are shown in Figure 5.4. The vertically exaggerated figure of the freshwater head and vertical cross-section at y=1500 m is shown also shown in Figure 5.4. Cones of depression for the wells can be seen in the same figure. The cones are more apparent for the wells far away from the cost since the optimal extraction rates found for these wells come out to be bigger.

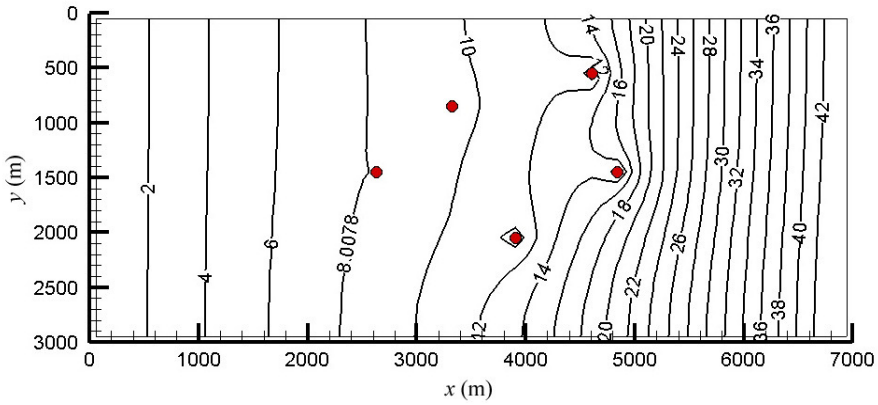
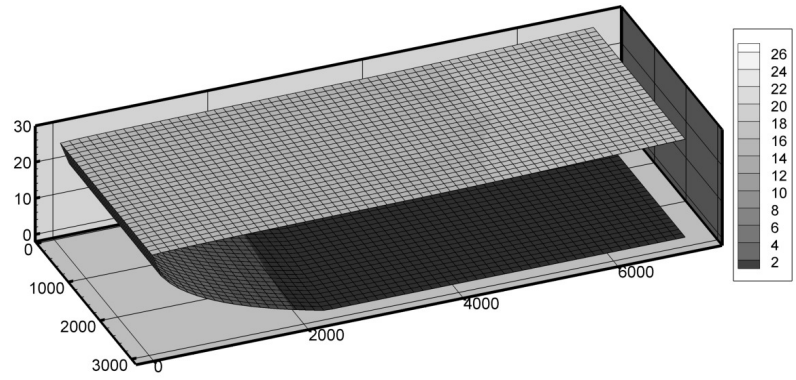
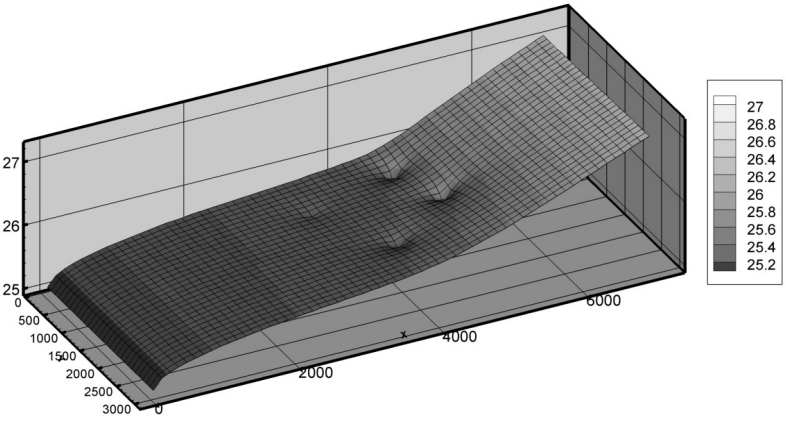


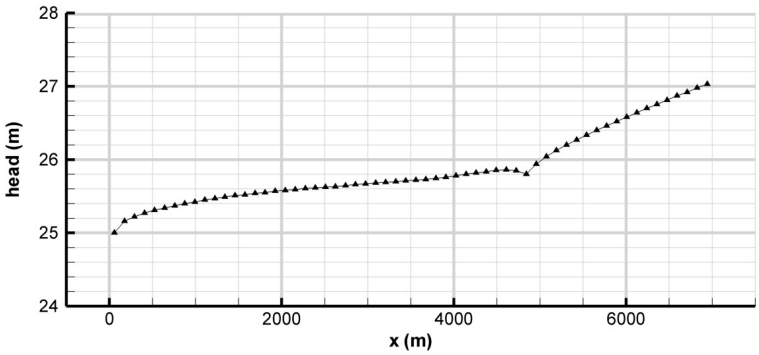
Figure 5.3 ϕ values corresponding to optimal solution



a)



b)



c)

Figure 5.4 a) Saltwater-freshwater interface for the optimal solution b) Freshwater zone vertically exaggerated c) Longitudinal cross-section of freshwater head at *y*=1500 m

The same optimization problem is then solved by using SA. For testing different initial points and stochastic effects involved in SA, the algorithm is run 5 times using different seed numbers.

Best solution vectors found by GA and SA are shown in Table 5.2. The results show that optimal results found by GA and SA are similar to the result found by LP (i.e. less than 0.1 %). Both results are better than the solution by Mantoglou (2003). The difference may be from the type of solver, parameters used for solver (i.e criteria for convergence), or hydraulic parameters that are not clearly presented in the paper.

Table 5.2 Optimal well rates for different algorithms

	Present Work			Mantoglou (2003)
Wells	GA	SA	LP	SQP
Q_{EXT1} (m^3/day)	58.82	52.39	51.44	436
Q_{EXT2} (m^3/day)	323.53	325.08	322.44	1089
Q_{EXT3} (m^3/day)	882.35	872.67	869.06	789
Q_{EXT4} (m^3/day)	1347.06	1348.20	1350.32	1149
Q_{EXT5} (m^3/day)	1476.47	1494.81	1500.0	483
Total (m^3/day)	4088.24	4093.15	4093.26	3945

The convergence characteristic of SA together with the optimal solution by GA is shown in Figure 5.5. From the figure, it can be seen that, SA required less number of function evaluations to reach optimum.

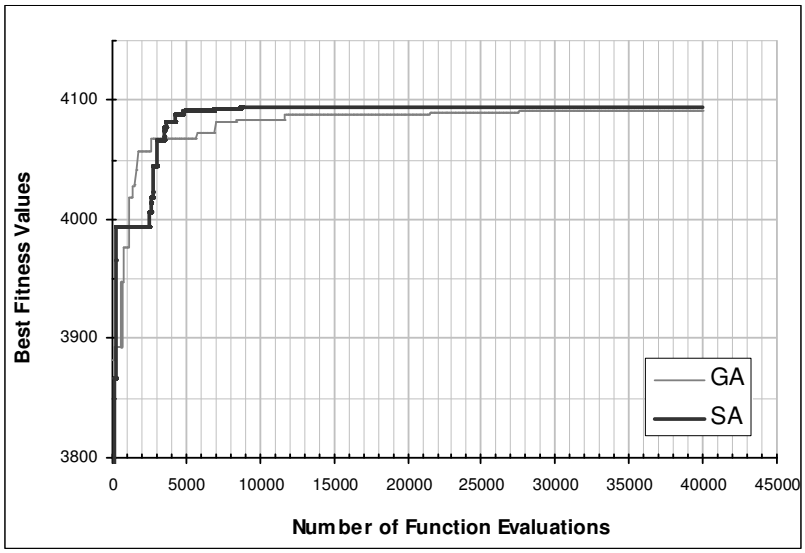


Figure 5.5 Best optimal results found by SA and GA

5.2 Injection Wells Problem

Utilizing injection wells is a well-known method used for the prevention of seawater intrusion in seawater treated sites. In common approach, series of wells are usually positioned on a line between seaside and the extraction area (Van Dam, 1999). This way, injections wells act as a physical barrier that prevents the further progress of saltwater. For the following problem, two different cases are studied. Case 1 represents the situation where locations of the wells are fixed and extraction and injection amounts are to be optimized. In Case 2, it is required to determine the optimal location of injection wells and corresponding pumpage rate for both injection and extraction wells. The location of the extraction wells are fixed for the present problem (i.e. to test the optimal locations and discharge of the injection wells for the case of extraction at specified locations) although their locations can also be optimized with the proposed model.

5.2.1 Case 1: Injection Wells Problem (Fixed Locations)

Three injection wells are added to the base model (Figure 5.6). The locations of the wells are fixed and their coordinates are listed in Table 5.3. The objective is to find the total maximum benefit without letting the seawater into extraction wells. Once again, 8.0078 m^2 represents the border where the seawater intrusion has reached. If the values at the extraction points are

lower than this value, the wells are accepted as intruded. The upper limit for the injection well rates is constrained to 500 m^3/day. Thus, the total amount of water that can be injected from three wells is 1500 m^3/day.

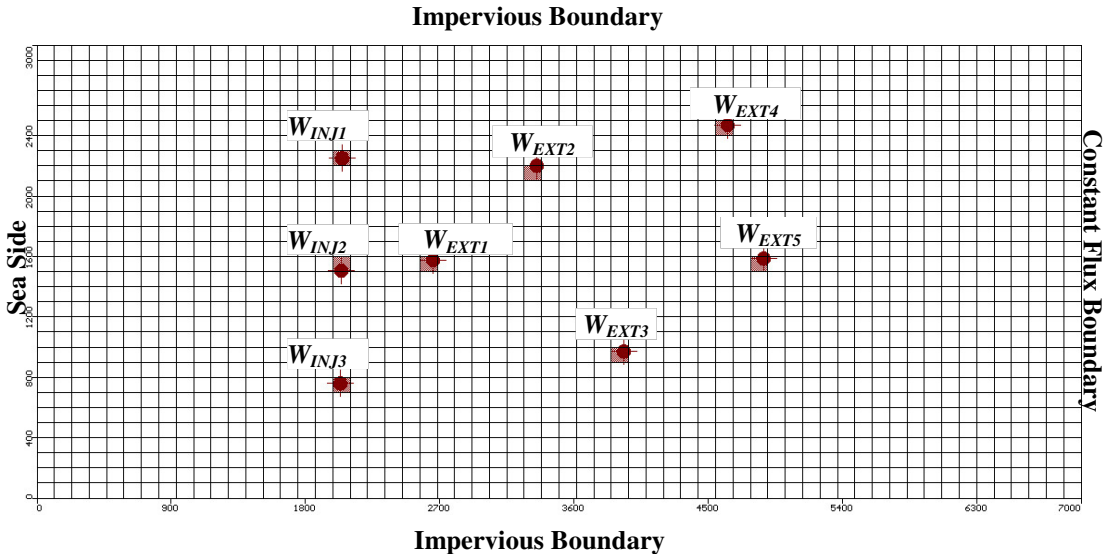


Figure 5.6 Finite difference representation of the model (where; W_{INJi} is the i^{th} injection well)

Table 5.3 Location of the injection wells

Extraction Wells	*x* (m)	*y* (m)
W_{INJ1}	2042	2250
W_{INJ2}	2042	1450
W_{INJ3}	2042	750

The formulation for the optimization problem is as follows;

The objective function,

$$Maximize\ f = \sum_{i=1}^{5} Q_{EXTi} - \alpha \sum_{j=1}^{3} Q_{INJj} \tag{5.5}$$

Set of constraints,

$$\phi_i \geq 8.0078\ \text{m}^2 \qquad i=1,2\ldots.5 \tag{5.6}$$

$$0 \leq Q_{EXTi} \leq 1500\ \text{m}^3/\text{day} \qquad i=1,2\ldots.5 \tag{5.7}$$

$$0 \leq Q_{INJj} \leq 500\ \text{m}^3/\text{day} \qquad j=1,2\ldots.3 \tag{5.8}$$

Unconstrained objective function,

$$Maximize\ f = \sum_{i=1}^{5} Q_{EXTi} - \alpha \sum_{j=1}^{3} Q_{INJj} - 10000 \times \sum_{i=1}^{5} (\max(0, 8.0078 - \phi_i))^2 \qquad (5.9)$$

The objective is to get the maximum benefit by extracting freshwater. Since the injected water which is utilized to prevent the seawater progress brings a cost to the solution, it is subtracted from the total amount of extraction. "α" constant, as explained previously, is the ratio of the economical value of injected water to the economical value of extracted water. It is initially taken as one (i.e. the economical value of injected water is same as the economical value of freshwater).

5.2.1.1 Discussion of the Results for Case 1: Injection Wells Problem (Fixed Locations)

From the optimal results shown in Table 5.4, it can be seen that for α=1, the amounts of water injected are almost zero (15.87 m^3/day for GA and 2.63 m^3/day for SA). This is concurrent with the fact that when the economical value of the extracted water is the same as the economical value of injected water, installing injection wells for the prevention of seawater intrusion becomes infeasible. However, in most situations the quality of injected water is cheaper than the extracted water (i.e. treated low cost wastewater or low quality freshwater obtained from saltwater) and α value would be less than one.

Best fitness values found (4072.37 for GA and 4077.68 for SA) are almost identical for GA and SA. However, combinations of discharge values for the wells, though close, are different. For the optimal results, difference between total extractions is less than 9 m^3/day, however the difference between extraction from each well can be greater than 110 m^3/day (i.e for W_{EXT5}).

It is also interesting to observe that although no or little injection is selected, the results are not as good as the results obtained from the extraction wells only problem (See Section 5.1). This shows that with the negative term included in the objective function due to new decision variables, the management problem became harder to solve for both optimizing algorithms.

Table 5.4 Optimal well rates (α=1.0)

	GA	SA	LP
Q_{EXT1} (m^3/day)	94.12	74.92	51.44
Q_{EXT2} (m^3/day)	376.47	373.97	322.44
Q_{EXT3} (m^3/day)	888.24	943.64	869.06
Q_{EXT4} (m^3/day)	1235.29	1307.12	1350.32
Q_{EXT5} (m^3/day)	1494.12	1380.66	1500.0
Q_{INJ1} (m^3/day)	0	1.30	0
Q_{INJ2} (m^3/day)	15.87	0.15	0
Q_{INJ3} (m^3/day)	0	1.18	0
Total Extracted Q (m^3/day)	4088.24	4080.31	4093.26
Total Injected Q (m^3/day)	15.87	2.63	0
Fitness Values	4072.37	4077.68	4093.26

In order to activate the injection wells, α is taken as 0.5 and the solutions are repeated once again using GA, SA and LP. The set of decision variables for the best solution (found among 20 runs for GA and 5 runs for SA) are given in Table 5.5. Since α<1, injection wells turned out to be feasible. The total amounts of injection for the solutions are close to the upper bounds for the injection wells, which is 1500 m^3/day. Total amount of extraction without injection (optimal solution found in Section 5.1.1) has also increased from 4093.26 m^3/day to 5007.01 m^3/day.

The best fitness values for GA and SA (4250.14 for GA and 4250.08 for SA) are almost identical and the corresponding decision variables for two solution are close with at most ≅6 % difference. The results are also very close to optimal solution found by LP.

Table 5.5 Optimal well rates (α=0.5)

	GA	SA	LP
Q_{EXT1} (m^3/day)	511.76	526.59	518.85
Q_{EXT2} (m^3/day)	541.18	547.01	533.63
Q_{EXT3} (m^3/day)	1029.41	1040.99	1025.21
Q_{EXT4} (m^3/day)	1405.88	1463.21	1429.32
Q_{EXT5} (m^3/day)	1500.00	1419.86	1500.00
Q_{INJ1} (m^3/day)	492.06	499.67	500.00
Q_{INJ2} (m^3/day)	484.13	497.46	500.00
Q_{INJ3} (m^3/day)	500.00	498.02	500.00
Total Extracted Q (m^3/day)	4988.23	4997.65	5007.01
Total Injected Q (m^3/day)	1476.19	1495.15	1500.00
Fitness Values	4250.14	4250.08	4257.01

Figure 5.7 shows the convergence characteristics for GA and SA, when α is taken as 0.5. It can be seen that SA converged to the optimal faster than GA. SA converges to best optimal around function evaluation number (from here on iteration number is used instead of number of function evaluations) 11000, while GA founds a similar result at around 29200. However, optimal result by GA at iteration number 11000 (fitness value of 4217.36) is only ≅1% less than the optimal result by SA (fitness value of 4250.08).

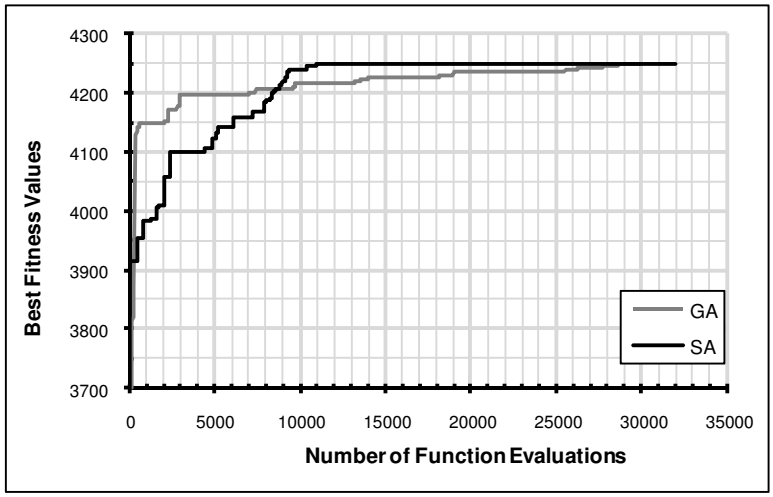


Figure 5.7 Best solution found by GA and SA

ϕ values corresponding to the best optimal solution and freshwater heads above seawater are shown in Figure 5.8 and Figure 5.9-a, respectively. ϕ_{toe} line which represents the toe of saltwater encorachment is not in front of the extraction wells (i.e Q_{EXT1}) but a little far away

from them. However, experiments with the model show that with a little increase in discharge value, ϕ_{toe} line jumps behind the extraction wells. This is in compliance with the results from Cheng et al. (2000). Vertical crosssection at y=1500 m is also shown in Figure 5.9-b.

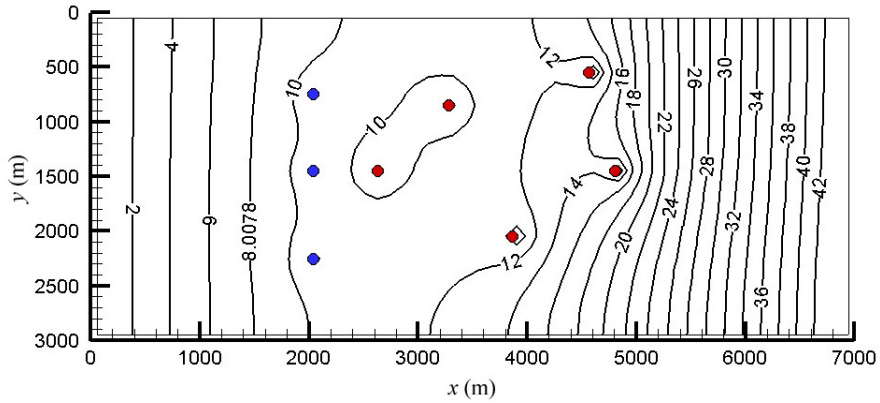
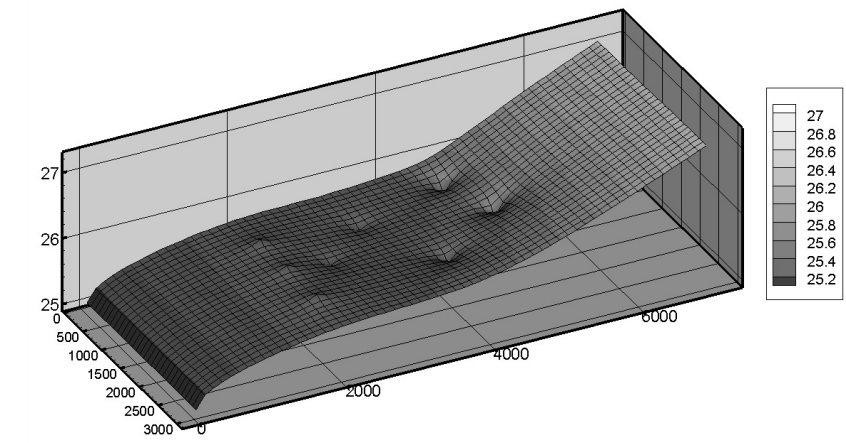
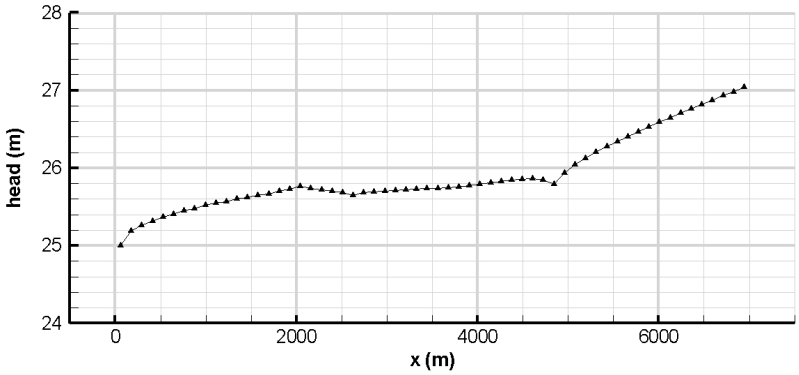


Figure 5.8 ϕ values corresponding to optimal solution GA



a)



b)

Figure 5.9 a) Freshwater head corresponding to optimal solution b) Longitudinal cross-section of freshwater head at y=1500 m

5.2.2 Case 2: Injection Wells Problem (Variable Locations)

In case 1, the locations of the injection wells were fixed. Including the locations as a decision variable in the optimization process may help to improve the results. The locations of the injection wells are constrained with the sea on the west, impervious boundary on the north and south and the first extraction well on the east direction (i.e. W_{EXT1}).

The objective function is as follows,

$$Maximize\ f = \sum_{i=1}^{5} Q_{EXTi} - \alpha \sum_{j=1}^{3} Q_{INJj} \tag{5.10}$$

Set of constraints,

$$\phi_i \geq 8.0078\ \text{m}^2 \qquad i=1,2\ldots.5 \tag{5.11}$$

$$0 \leq Q_{EXTi} \leq 1500\ \text{m}^3/\text{day} \qquad i=1,2\ldots.5 \tag{5.12}$$

$$0 \leq Q_{INJj} \leq 500\ \text{m}^3/\text{day} \qquad j=1,2\ldots.3 \tag{5.13}$$

$$0 \leq x_{INJj} \leq 20 \qquad j=1,2\ldots.3 \tag{5.14}$$

$$0 \leq y_{INJj} \leq 30 \qquad j=1,2\ldots.3 \tag{5.15}$$

Unconstrained objective function,

$$Maximize\ f = \sum_{i=1}^{5} Q_{EXTi} - \alpha \sum_{j=1}^{3} Q_{INJj} - 10000 \times \sum_{i=1}^{5} (\max(0, 8.0078 - \phi_i))^2 \tag{5.16}$$

The optimization formulation is tested for α= 0.5.

5.2.2.1 Discussion of the Results for Case 2: Injection Wells Problem (Variable Locations)

The best optimal results found are shown in Table 5.6. Again, the best fitness values from GA and SA are almost same (fitness value of 4358.50 for GA and 4356.77 for SA). For both solutions, it is seen that injection wells are gathered in the middle in *y* direction and to the farthest point from the sea in *x* direction (i.e. the upper bound for the *x* constraint for injection well locations). This is expected, since injection wells and canals are reported to be not effective when placed far too seaward from pumping wells (Cheng and Ouazar, 1999). For the discharge values, SA chooses a solution where total extraction (5000.10 m^3/day) is less than the total extraction found by GA solution (5052.94 m^3/day). In proportion,

injection amount is also less for the solution with SA (1286.66 m^3/day with respect to the 1388.89 m^3/day for GA). Best fitness values are slightly worse than the solution found by MIP (i.e. less than 2 %). In addition, both SA and GA fail to find the optimal solution for which the location for the three injection wells is same (i.e. x_{inj}=15 and y_{inj}=20).

Table 5.6 Optimal well rates and locations (α=0.5)

	GA	SA	MIP
Q_{EXT1} (m^3/day)	700.00	637.99	740.66
Q_{EXT2} (m^3/day)	652.94	537.59	512.53
Q_{EXT3} (m^3/day)	1170.59	1068.11	1003.64
Q_{EXT4} (m^3/day)	1500.00	1496.08	1417.87
Q_{EXT5} (m^3/day)	1029.41	1260.33	1500.00
Q_{INJ1} (m^3/day)	468.25	293.47	500.00
Q_{INJ2} (m^3/day)	420.63	499.70	500.00
Q_{INJ3} (m^3/day)	500.00	493.50	500.00
x_{INJ1}	20	20	20
y_{INJ1}	14	15	15
x_{INJ2}	20	20	20
y_{INJ2}	13	13	15
x_{INJ3}	20	20	20
y_{INJ3}	13	16	15
Total Extracted Q (m^3/day)	5052.94	5000.10	5174.70
Total Injected Q (m^3/day)	1388.89	1286.66	1500.00
Fitness Values	4358.50	4356.77	4424.70

*Locations are given as the coordinates of cells in finite difference representation.

The ϕ distribution and the locations corresponding to best optimal solution are shown in Figure 5.10. Three-dimensional representation of freshwater head above seawater and a cross-section from the same graph are shown in Figure 5.11-a and Figure 5.11-b, respectively.

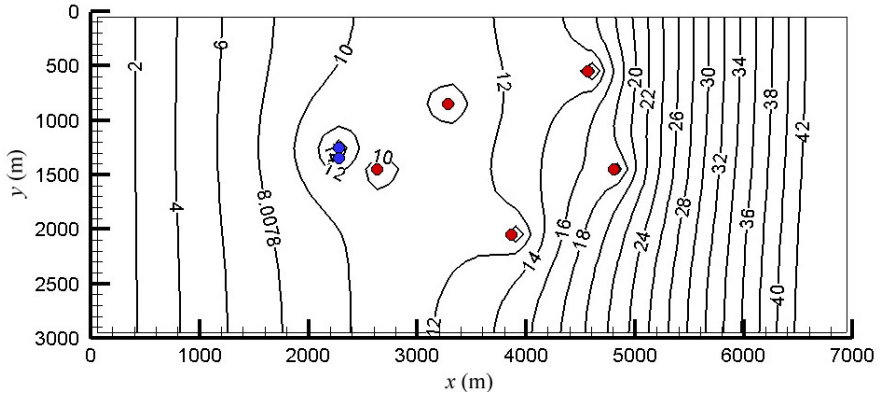
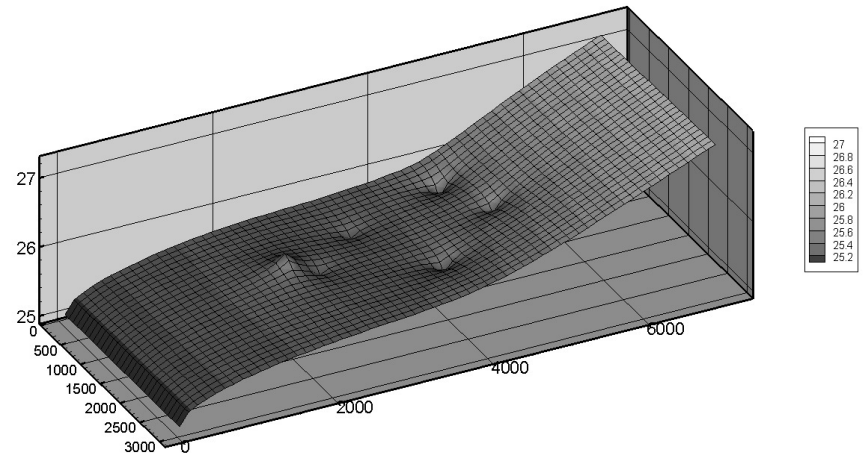
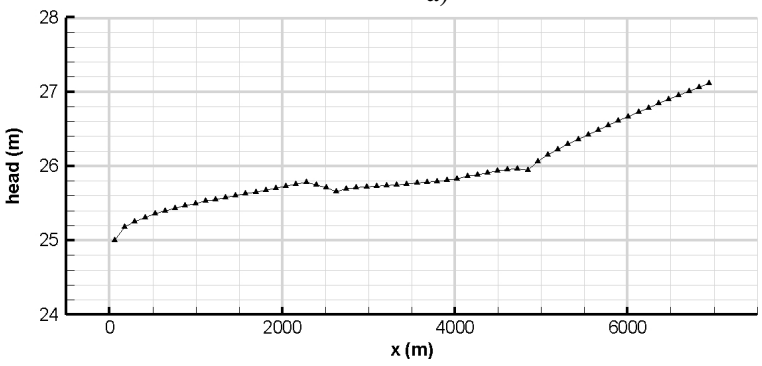


Figure 5.10 ϕ values corresponding to optimal solution



a)



b)

Figure 5.11 a) Freshwater head corresponding to optimal solution b) Longitudinal cross-section of freshwater head at *y*=1500 m

5.2.3 An Improvement for the Solution Technique: Alternating Constraints Method

An individual in a population is a representation of all the unknowns that are to be optimized. As briefly discussed in previous sections, for the solution with traditional GA, all the decision variables (i.e. including both the discharge values and location of injection wells for the current problem), are to be encoded as series of binary strings. As the length of individuals are increased, the number of solutions among which GA tries to find the optimal (search area) increases. In addition, increasing number of decision variables results in increasing complexity in solution space, which makes the problem hard to solve for the optimization methods. In this study, a method called Alternating Constraints Method is introduced where discharge values and the locations are optimized one at a time in an iterative manner.

Procedure of the method is as follows:

Step 1: Fix the location of the injection wells to arbitrary locations within the constraint limits (Eqns. 5.14 and 5.15). Run GA to find the optimal discharge of extraction and injection wells (Eqn. 5.16). ϕ_i and discharge values are constrained by Eqns. 5.11, 5.12 and 5.13, respectively.

Step 2: Fix the well rates (both extraction and injection) with the optimal rates found in Step 1. Optimize the location of the injection wells by maximizing the ϕ differences in the control locations by running GA once again (Eqn. 5.17). The locations of the injection wells are again constrained by Eqns. 5.14 and 5.15.

$$Maximize\ f = \sum_{i=1}^{m} \max(0, \phi_i - \phi_{toe}) \tag{5.17}$$

where, m is the number of extraction wells, for which the ϕ values are checked.

Step 3: Repeat Step 1 once again, this time by fixing the well locations with the optimal locations found in step 2. Flowchart of the method is given in Figure 5.12.

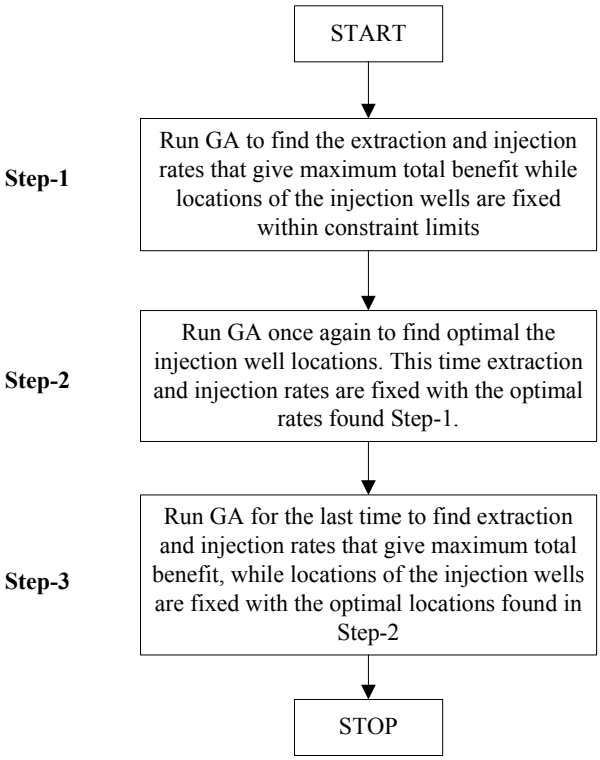


Figure 5.12 The flowchart for the current method

5.2.3.1 Discussion of the Results for Case 2: Injection Wells Problem (Variable Locations) using Alternating Constraints Method

The method introduced is applied to Case 2: Injection Well Problem (Variable Locations). Convergence character for the best optimal solution found is shown in Figure 5.13. From the figure, it can be seen that in Step 1, the algorithm converges to a value of 4332.16. However, after step 2 where the locations of the wells are to be optimized, the algorithm converges to a better solution, 4402.25 in Step 3 of proposed method.

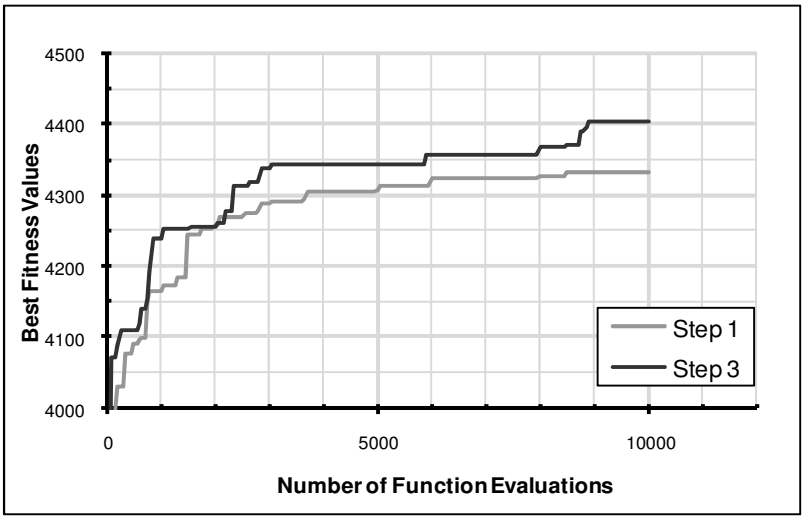


Figure 5.13 Best fitness values corresponding to optimal solution for Alternating Constraints Method

The results from different optimization solvers used are shown in Table 5.7. Among the randomized search algorithms, the best optimal result (i.e. 4402.25) is found by the Alternating Constraints Method. The worst solution by Alternating Constraints Method is better with respect to the worst solution by GA. Average and worst solutions for SA are higher, which shows a smoother distribution for the solutions by SA.

Table 5.7 The maximum, minimum and average of the optimal solutions for the Injection Well Problem (Variable Locations), α=0.5 (Among 20 runs for GA and 5 runs for SA)

	Best Solution	Worst Solution	Average
Alternating Constraints Method	4402.25	4134.08	4256.15
GA	4358.50	4087.12	4261.99
SA	4356.77	4279.18	4319.93
MIP	4424.70	-	-

For the solution with MIP, each potential location for the injection wells is an additional integer variable in the optimization formulation. The number of integer variables for the current example is 9000 (50 (number of columns in x direction) $\times$ 60 (number of rows in y direction) $\times$ 3 (number of injection wells)). Maximum number of integer variables that can be solved with the trial version of LINDO is only 50. Thus, the search domain is divided into smaller zones and the solution method is repeated for each zone. It must be noted that this approach does not guarantee an optimal solution, considering that optimal locations may cover different zones.

ϕ distribution for the best optimal solution is shown in Figure 5.14. Freshwater head distribution for the coastal aquifer and a cross-section in *y*-dimension is shown in Figure 5.15.

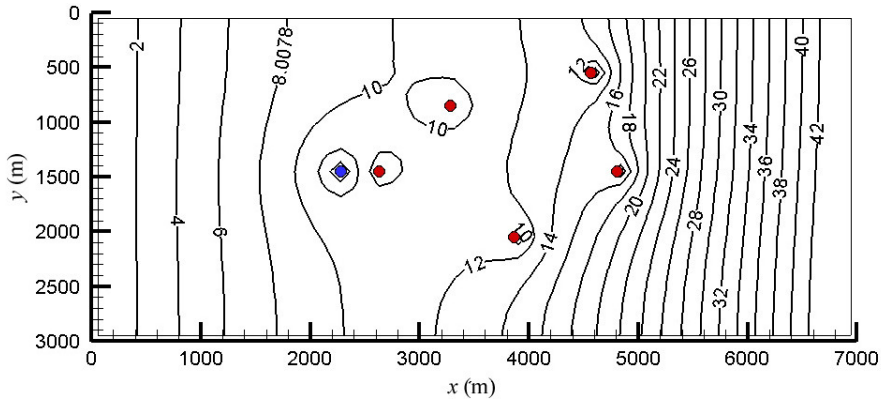
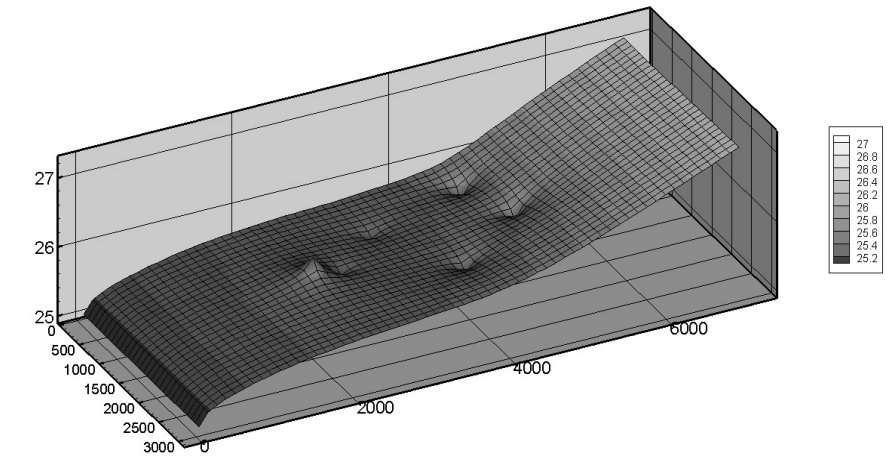
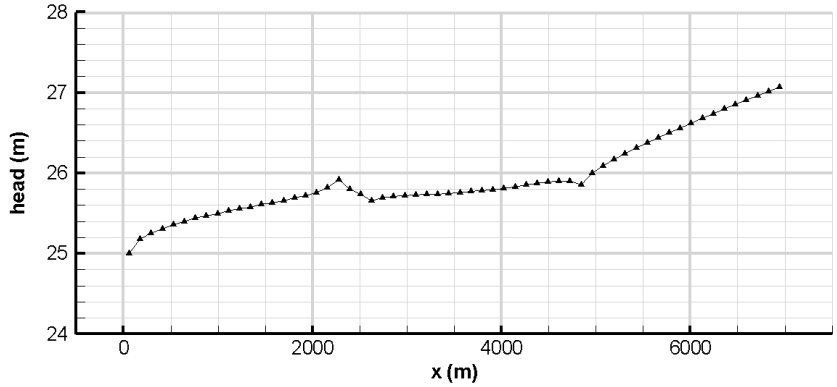


Figure 5.14 ϕ values corresponding to optimal solution GA



a)



b)

Figure 5.15 a) Freshwater head corresponding to optimal solution b) Longitudinal cross-section of freshwater head at *x*=1500 m

5.2.4 Summary of the Results

For the injection well problem, the solution sets corresponding to the best optimal solutions for the case with fixed locations and the case with variable locations are shown in Table 5.8. As discussed in Section 5.2.1, optimal values found when the location of the wells are fixed are almost identical (i.e. values changing from 4250.08 to 4257.01). When the locations of the injection wells are included in the optimization process, the fitness value has increased from 4250.14 to 4358.50, for the solution with GA. Total extraction amount has increased from 4988.23 m^3/day to 5052.94 m^3/day. When Alternating Constraints Method is applied, fitness value has increased to 4402.25 and total extraction amount to 5141.18 m^3/day.

5.3 Canal Problem

As to form a barrier for seawater intrusion, instead of injecting from a point source (well), water can be articially recharged by using surface spreading (i.e. water canal) for unconfined aquifers (Todd, 1980). Since groundwater flow is assumed to be horizontal, the flow dynamics for the model will be independent of height of the canal and canal is modeled as an additional source of recharge.

For the canal problem, two different management models are formulated for both of which the objective functions are same, while decision variables and constraints are different. The management models seek for the maximum amount of extraction while recharge is provided by a canal.

For the first case, the location and length of canal is to be optimized while recharge rate for the canal is fixed. For the second case, the location and recharge rate for the canal is to be optimized while length of the canal is fixed.

5.3.1 Case 1: Canal Problem (Variable Location and Length)

The objective is to use the maximum groundwater potential while there is no seawater intrusion. ϕ value of 8.0078 m^2 represents the border where the seawater intrusion has reached. The canal utilized in the study is parallel to the coastline and extends as a straight line in y direction. Length and location of the canal together with the total amount of

Table 5.8 Best optimal results found for Injection Well Problem, α=0.5 (Bolds are values that are optimized, the other values are fixed for the related solution)

	Optm. Extr. and Inj.Rates (Loc. of the Inj. Wells are Fixed)*	Optm. Extr. and Inj.Rates (Loc. of the Inj. Wells are Fixed)*	Optm. Extr. and Inj.Rates (Loc of the Inj. Wells are Fixed)*	Optm. Extr. and Inj. Rates together with the Optm. Loc. of the Inj. Wells**	Optm. Extr. and Inj. Rates together with the Optm. Loc. of the Inj. Wells**	Optm. Extr. and Inj. Rates together with the Optm. Loc. of the Inj. Wells**	Optm. Extr. and Inj. Rates together with the Optm. Loc. of the Inj. Wells**
	GA	SA	LP	GA	SA	MIP	Alternating Constraints Method
Q_{EXT1} (m^3/day)	**511.76**	**526.59**	**518.85**	**700.00**	**637.99**	**740.66**	**758.82**
Q_{EXT2} (m^3/day)	**541.18**	**547.01**	**533.63**	**652.94**	**537.59**	**512.53**	**564.71**
Q_{EXT3} (m^3/day)	**1029.41**	**1040.99**	**1025.21**	**1170.59**	**1068.11**	**1003.64**	**1052.94**
Q_{EXT4} (m^3/day)	**1405.88**	**1463.21**	**1429.32**	**1500.00**	**1496.08**	**1417.87**	**1452.94**
Q_{EXT5} (m^3/day)	**1500.00**	**1419.86**	**1500.00**	**1029.41**	**1260.33**	**1500.00**	**1311.77**
Q_{INJ1} (m^3/day)	**492.06**	**499.67**	**500.00**	**468.25**	**293.47**	**500.00**	**500.00**
Q_{INJ2} (m^3/day)	**484.13**	**497.46**	**500.00**	**420.63**	**499.70**	**500.00**	**476.19**
Q_{INJ2} (m^3/day)	**500.00**	**498.02**	**500.00**	**500.00**	**493.50**	**500.00**	**500.00**
x_{INJ1}	18	18	18	**20**	**20**	**20**	**20**
y_{INJ1}	8	8	8	**14**	**15**	**15**	**15**
x_{INJ2}	18	18	18	**20**	**20**	**20**	**20**
y_{INJ2}	15	15	15	**13**	**13**	**15**	**15**
x_{INJ3}	18	18	18	**20**	**20**	**20**	**20**
y_{INJ3}	23	23	23	**13**	**16**	**15**	**15**
Total Extracted Q (m^3/day)	**4988.23**	**4997.65**	**5007.01**	**5052.94**	**5000.10**	**5174.70**	**5141.18**
Total Injected Q (m^3/day)	**1476.19**	**1495.15**	**1500.00**	**1388.89**	**1286.66**	**1500.00**	**1476.19**
Fitness	**4250.14**	**4250.08**	**4257.01**	**4358.50**	**4356.77**	**4424.70**	**4402.25**

* Optimal Extraction and Injection Rates (Locations of the Injection Wells are Fixed)

** Optimal Extraction and Injection Rates together with the Optimal Locations of the Injection Wells

extraction from five wells are to be optimized. l_{can} is the number of grids that canal occupies in y direction. Upper limit for l_{can} is 10. Recharge rate is fixed to 1.2857×10^{-2} $m^3/day/m^2$ for which the total amount of discharge for the maximum length of canal is equal to 1500 m^3/day. Note that this amount is equal to the total upper bound for the injection wells studied in the previous problem. This is to make the comparisons between the problems easier. β, which is the ratio of the economical value of recharged canal water to the economical value of extracted water, is taken as 0.5.

The objective function is as follows,

$$Maximize\ f = \sum_{i=1}^{m} Q_{EXTi} - \beta.\Delta x.\Delta y.l_{CAN}.R_{CAN} \tag{5.18}$$

Set of constraints,

$$\phi_i \geq 8.0078\ m^2 \qquad i=1,2....5 \tag{5.19}$$

$$0 \leq Q_{EXTi} \leq 1500\ m^3/day \qquad i=1,2....5 \tag{5.20}$$

$$0 \leq l_{CAN} \leq 10 \tag{5.21}$$

$$1 \leq x_{CAN} \leq 20 \tag{5.22}$$

$$1 \leq y_{CAN} \leq 30 \tag{5.23}$$

Unconstrained objective function,

$$Maximize\ f = \sum_{i=1}^{m} Q_{EXTi} - \beta.\Delta x.\Delta y.l_{CAN}.R_{CAN} - 10000\times\sum_{i=1}^{m}(\max(0, 8.0078-\phi_i))^2 \tag{5.24}$$

5.3.1.1 Application of Alternating Constraints method to Case 1: Canal Problem (Variable Location and Length)

The method introduced in Section 5.2.3, where locations and recharge values are optimized in an iterative manner, is applied to Case 1: Canal Problem.

Procedure of the method is as follows:

Step 1: Fix the location of the canal to a random location within the constraint limits (Eqns. 5.22 and 5.23). Run GA to find the optimal discharge for extraction wells and optimal length for canal. The objective function (Eqn. 5.24) used to find the extraction rates, ϕ_i values and length of the canal are constrained by Eqns. 5.19 and 5.20.

Step 2: Fix the extraction rates and length with the optimal rates found in Step 1. Optimize the location of the canal by maximizing the ϕ differences in the control locations by running GA once again (Eqn. 5.25). The location of the canal is again constrained by Eqns. 5.22 and 5.23.

$$Maximize\ f = \sum_{i=1}^{m} \max(0, \phi_i - \phi_{toe}) \tag{5.25}$$

Step 3: Repeat Step 1 once again, this time by fixing the location of the canal with the optimal location found in Step 2.

The method is applied to Case 1: Canal Problem (Variable Location and Length). Runs are repeated for 20 times with different seeds to examine the effects of stochastic effects involved in solution.

The best optimal solution found by the Alternating Constraints method, GA and SA is shown in Table 5.9. In all solutions, optimum length of the canal is found as 10 grids. This is the upper limit for the length of the canal for which recharge is fully applied. In all solutions, *x*-coordinate of the canal is found as 20 and *y*-coordinate is found around to the middle of the aquifer. The fitness values from different algorithms are almost identical, the result from Alternating Constraints Method (with a fitness value of 4397.06) is the best one.

Table 5.9 Optimal well rates for β=0.5 (Bold values are optimized values, the other values are fixed)

	Alternating Constraints Method	GA	SA
Q_{EXT1} (m^3/day)	**676.47**	**658.82**	**640.33**
Q_{EXT2} (m^3/day)	**582.35**	**605.88**	**660.54**
Q_{EXT3} (m^3/day)	**1017.65**	**1035.29**	**1175.94**
Q_{EXT4} (m^3/day)	**1376.47**	**1388.24**	**1360.45**
Q_{EXT5} (m^3/day)	**1494.12**	**1452.94**	**1258.74**
Canal Length	**10**	**10**	**10**
Canal x-coordinate	**20**	**20**	**20**
Canal y-coordinate	**9**	**8**	**5**
Canal Recharge Rate (m^3/day/m^2)	1.2857×10^{-2}	1.2857×10^{-2}	1.2857×10^{-2}
Total Extracted Q (m^3/day)	**5147.06**	**5141.18**	**5096.00**
Total Recharged Q (m^3/day)	**1500.00**	**1500.00**	**1500.00**
Fitness	**4397.06**	**4391.18**	**4346.00**

The ϕ distribution is shown in Figure 5.16. The potential line of 8.0078 m^2 is in front of the extraction wells, which shows that seawater has not progressed behind the extraction wells for the solution.

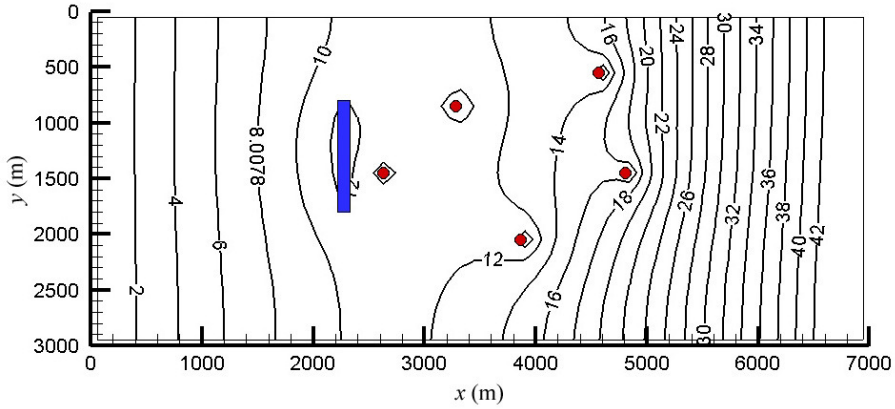


Figure 5.16 ϕ values corresponding to optimal solution

Figure 5.17 shows the freshwater head distribution above seawater for the coastal aquifer. The location of the canal can be seen as a line shaped depression in front of the extraction wells.

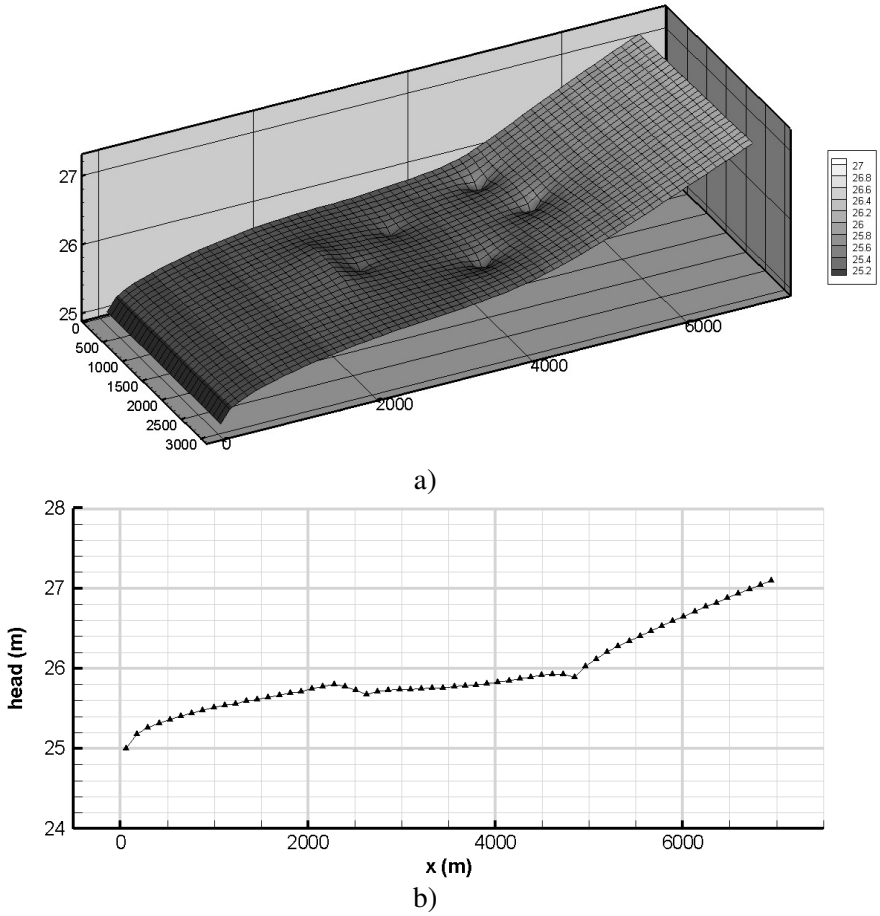


Figure 5.17 a) Freshwater head corresponding to optimal solution. b) Longitudinal cross-section of freshwater head at x=1500 m

5.3.2 Case 2: Canal Problem (Variable Location and Recharge)

Similar to Case 1, the objective is to use the maximum groundwater potential while there is no seawater intrusion. The location and recharge value of the canal (instead of length of the canal in Case 1) are to be optimized together with the total extraction from five wells. The upper limit for the recharge of the canal is constrained to 1.2857×10^{-2} $m^3/day/m^2$. Canal length is fixed to 10 grids in y direction (as found in Case 1) for which the total amount of discharge for the maximum recharge of the canal is equal to 1500 m^3/day.

The objective function is as follows,

$$Maximize\ f = \sum_{i=1}^{m} Q_{EXTi} - \beta.\Delta x.\Delta y.l_{CAN}.R_{CAN} \tag{5.26}$$

Set of constraints,

$$\phi_i \geq 8.0078 \text{ m}^2 \qquad i=1,2\ldots.5 \tag{5.27}$$

$$0 \leq Q_{EXTi} \leq 1500 \text{ m}^3/\text{day} \qquad i=1,2\ldots.5 \tag{5.28}$$

$$0 \leq R_{CAN} \leq 1.2857 \times 10^{-2} \text{ m}^3/\text{day/m}^2 \tag{5.29}$$

$$1 \leq x_{CAN} \leq 20 \tag{5.30}$$

$$1 \leq y_{CAN} \leq 30 \tag{5.31}$$

Unconstrained objective function,

$$Maximize\ f = \sum_{i=1}^{m} Q_{EXTi} - \beta.\Delta x.\Delta y.l_{CAN}.R_{CAN} - 10000 \times \sum_{i=1}^{m} (\max(0, 8.0078 - \phi_i))^2 \tag{5.32}$$

The optimization formulation is tested for β=0.5.

5.3.2.1 Application of Alternating Constraints Method on Canal Problem (Variable Location and Recharge)

Procedure of the method is as follows:

Step 1: Fix the location of the canal to a random location within the constraint limits (Eqns. 5.30 and 5.31). Run GA to find the optimal discharge for extraction wells and optimal recharge for canal (Eqn. 5.32). The ϕ_i values, extraction rates, and recharge value for the canal are constrained by Eqns. 5.27, 5.28 and 5.29, respectively.

Step 2: Fix the extraction and recharge rates with the optimal rates found in Step 1. Optimize the location of the canal by maximizing the ϕ differences in the control locations by running GA once again (Eqn. 5.33). The location of the canal is again constrained by Eqns. 5.30 and 5.31.

$$Maximize\ f = \sum_{i=1}^{m} \max(0, \phi_i - \phi_{toe}) \tag{5.33}$$

Step 3: Repeat step 1 once again, this time by fixing the location of the canal with the optimal location found in step 2.

The method is applied to Case 2: Canal Problem (Variable Location and Recharge). Again the runs are repeated for 20 times. The problem is also solved with SA for comparison purposes.

The best optimal results are shown in Table 5.10. The results are again almost identical for different algorithms. For the optimal solution, canals locate themselves on the upper limit of the x direction farthest point from the coast and around to the middle in y direction. Recharge rates are found close to 1.2857×10^{-2} $m^3/day/m^2$ for which the total amount of recharge is 1500 m^3/day. This is the upper limit for the recharge value.

Table 5.10 Optimal well rates for β=0.5 (Bold values are optimized values, the other values are fixed)

	Alternating Constraints Method	GA	SA
Q_{EXT1} (m^3/day)	**700.00**	**805.88**	**668.01**
Q_{EXT2} (m^3/day)	**670.59**	**817.65**	**572.63**
Q_{EXT3} (m^3/day)	**1152.94**	**1235.29**	**1030.37**
Q_{EXT4} (m^3/day)	**1417.65**	**1329.41**	**1418.44**
Q_{EXT5} (m^3/day)	**1182.35**	**876.47**	**1455.09**
Canal Length	10	10	10
Canal x-coordinate	**20**	**20**	**20**
Canal y-coordinate	**8**	**10**	**9**
Canal Recharge Rate ($m^3/day/m^2$)	**1.2857×10^{-2}**	**1.2857×10^{-2}**	**1.2832×10^{-2}**
Total Extracted Q (m^3/day)	**5123.53**	**5064.71**	**5144.54**
Total Recharged Q (m^3/day)	**1500.00**	**1500.00**	**1497.01**
Fitness	**4373.53**	**4314.71**	**4396.03**

Potential distribution for the best solution found is shown in Figure 5.18. The freshwater distribution above seawater is shown in Figure 5.19.

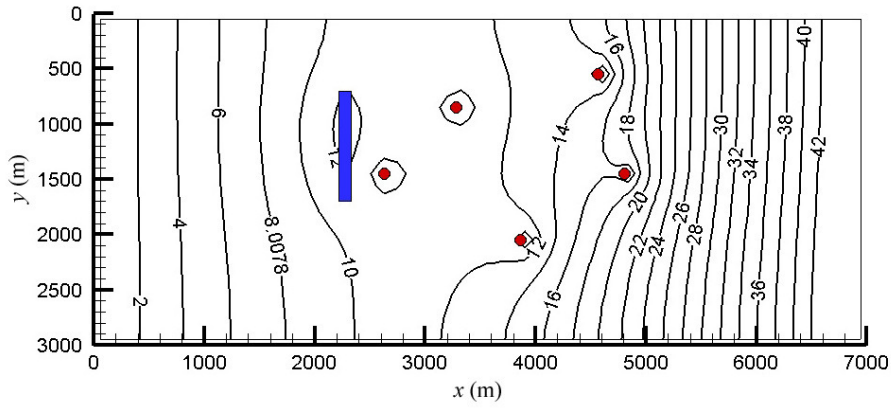
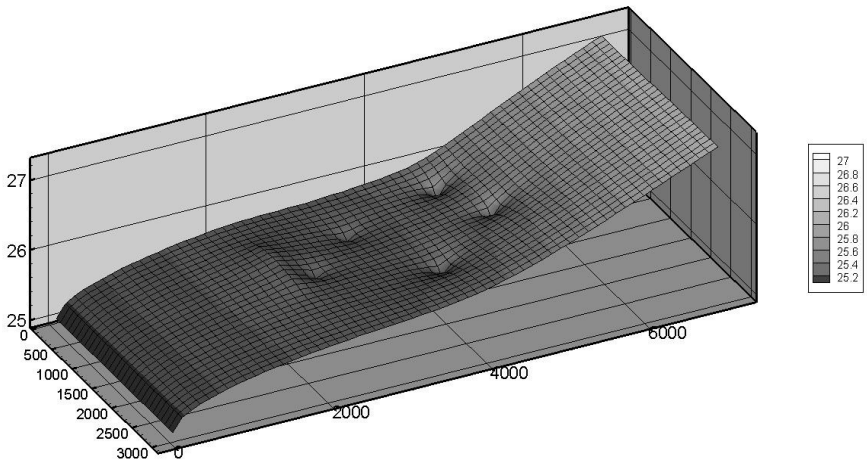
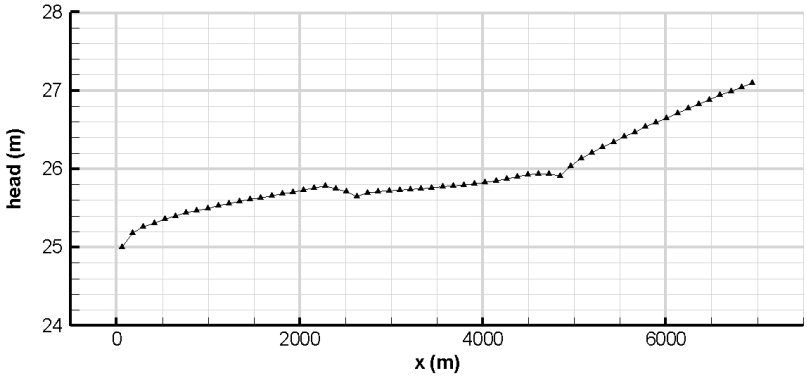


Figure 5.18 ϕ values corresponding to optimal solution GA



a)



b)

Figure 5.19 a) ϕ values corresponding to optimal solution. b) Longitudinal cross-section of freshwater head at x=1500 m

5.3.3 Summary of the Results

In Table 5.11, the results from Case 1: Canal problem (Variable Location and Length) and Case 2: Canal problem (Variable Location and Recharge) are put side by side. The results give a similar picture especially for the locations, length and recharge values for the canal. The results from the two different cases turn out to be identical, for which the different decision variables are optimized but similar results are found for the same objective.

Table 5.11 Optimal well rates for β=0.5 (Bold values are optimized values, the other values are fixed)

	Optimal Canal Location and Length	Optimal Canal Location and Length	Optimal Canal Location and Length	Optimal Canal Location and Recharge Rate	Optimal Canal Location and Recharge Rate	Optimal Canal Location and Recharge Rate
	Alternating Constraints Method	GA	SA	Alternating Constraints Method	GA	SA
Q_{EXT1} (m^3/day)	**676.47**	**658.82**	**640.33**	**700.00**	**805.88**	**668.01**
Q_{EXT2} (m^3/day)	**582.35**	**605.88**	**660.54**	**670.59**	**817.65**	**572.63**
Q_{EXT3} (m^3/day)	**1017.65**	**1035.29**	**1175.94**	**1152.94**	**1235.29**	**1030.37**
Q_{EXT4} (m^3/day)	**1376.47**	**1388.24**	**1360.45**	**1417.65**	**1329.41**	**1418.44**
Q_{EXT5} (m^3/day)	**1494.12**	**1452.94**	**1258.74**	**1182.35**	**876.47**	**1455.09**
Canal Length	**10**	**10**	**10**	10	10	10
Canal *x*-coordinate	**20**	**20**	**20**	**20**	**20**	**20**
Canal *y*-coordinate	**9**	**8**	**5**	**8**	**10**	**9**
Canal Recharge Rate(m^3/day/m^2)	1.2857×10^{-2}	1.2857×10^{-2}	1.2857×10^{-2}	$\mathbf{1.2857\times10^{-2}}$	$\mathbf{1.2857\times10^{-2}}$	$\mathbf{1.2832\times10^{-2}}$
Total Extracted Q (m^3/day)	**5147.06**	**5141.18**	**5096.00**	**5123.53**	**5064.71**	**5144.54**
Total Injected Q (m^3/day)	**1500.00**	**1500.00**	**1500.00**	**1500.00**	**1500.00**	**1497.01**
Fitness	**4397.06**	**4391.18**	**4346.00**	**4373.53**	**4314.71**	**4396.03**

5.4 Injection Wells & Canal Problem

A management problem is designed where two different seawater prevention methods discussed in previous problems (injection and canal) and related decision variables (locations and recharge values) are included in a single problem. Total amount of recharge (by injection wells and canal) are constrained so that it is shared between injection wells and canal. This is done by using penalty method. If the total amount is exceeded, a penalty is applied to the objective function, proportional to the degree of violation.

The objective function is as follows,

$$Maximize\ f = \sum_{i=1}^{m} Q_{EXTi} - \alpha \sum_{j=1}^{3} Q_{INJj} - \beta.\Delta x.\Delta y.l_{CAN}.R_{CAN} \tag{5.34}$$

Set of constraints,

$$\phi_i \geq 8.0078\ \text{m}^2 \qquad i=1,2....5 \tag{5.35}$$

$$0 \leq Q_{EXTi} \leq 1500\ \text{m}^3/\text{day} \qquad i=1,2....5 \tag{5.36}$$

$$0 \leq Q_{INJj} \leq 500\ \text{m}^3/\text{day} \qquad j=1,2....3 \tag{5.37}$$

$$0 \leq x_{INJj} \leq 20 \qquad j=1,2....3 \tag{5.38}$$

$$0 \leq y_{INJj} \leq 30 \qquad j=1,2....3 \tag{5.39}$$

$$0 \leq R_{CAN} \leq 1.2857 \times 10^{-2}\ \text{m}^3/\text{day}/\text{m}^2 \tag{5.40}$$

$$1 \leq x_{CAN} \leq 20 \tag{5.41}$$

$$1 \leq y_{CAN} \leq 30 \tag{5.42}$$

$$0 \leq \sum_{j=1}^{3} Q_{INJj} + l_{CAN}.\Delta x.\Delta y.R_{CAN} \leq 1500\ \text{m}^3/\text{day} \tag{5.43}$$

Unconstrained objective function,

$$\begin{aligned} Maximize\ f = & \sum_{i=1}^{m} Q_{EXTi} - \alpha \sum_{j=1}^{3} Q_{INJj} - \beta . l_{CAN} . \Delta x . \Delta y . R_{CAN} \\ & - 10000 \times \sum_{i=1}^{m} (\max(0, 8.0078 - \phi_i))^2 \\ & - 10000 \times (\max(0, \sum_{j=1}^{3} Q_{INJj} + l_{CAN} . \Delta x . \Delta y . R_{CAN} - 1500))^2 \end{aligned} \tag{5.44}$$

The optimization formulation is tested for the case with α and β =0.5.

5.4.1 Discussion of the Results for Injection Wells & Canal Problem

The best optimal results found by GA and SA are shown in Table 5.12. Solution by SA chooses to recharge from the canal instead of injecting from the wells. Solution by GA, on the other hand, reaches a mix solution, where some of the water is injected from the wells while remaing is recharged from the canals. Similar fitness values indicate that either solution gives similar benefits. This is expected, since optimal fitness values for injection well only case (4358.50, see Section 5.2) was close to the optimal solution for canal only case (4391.18, see Section 5.3) for the solution with GA. It is also noted that optimal results given in Table 5.12 are also similar to the previously found optimals. A choice between two options (canal or injection well) would require other criterias that are not involved in our solution (i.e. the cost of installation, sustainability questions or environmental affects)

Table 5.20 shows the convergence behaviour of SA and GA. Although, SA converged to a slightly better value, GA found a near optimal value faster.

Table 5.12 Optimal decision variables for the Injection Wells & Canal Problem for α and β=0.5 (Bolds are optimized values, the others are fixed values)

	GA	SA
Q_{EXT1} (m^3/day)	**558.82**	**671.76**
Q_{EXT2} (m^3/day)	**558.82**	**578.06**
Q_{EXT3} (m^3/day)	**1005.88**	**1029.03**
Q_{EXT4} (m^3/day)	**1435.29**	**1372.34**
Q_{EXT5} (m^3/day)	**1488.24**	**1495.15**
Q_{INJ1} (m^3/day)	**277.78**	**0.01**
Q_{INJ2} (m^3/day)	**325.40**	**0.00**
Q_{INJ3} (m^3/day)	**15.87**	**0.00**
x_{INJ1}	**20**	**5**
y_{INJ1}	**17**	**20**
x_{INJ2}	**20**	**12**
y_{INJ2}	**8**	**6**
x_{INJ3}	**20**	**15**
y_{INJ3}	**20**	**5**
Canal Length (m)	10	10
Canal x-coordinate	**20**	**20**
Canal y-coordinate	**8**	**9**
Canal Recharge Rate (m^3/day/m^2)	**0.6537×10^{-2}**	**1.2856×10^{-2}**
Total Extracted Q (m^3/day)	**5047.06**	**5146.35**
Total Recharged Q (m^3/day)	**1382.38**	**1499.99**
Fitness	**4355.87**	**4396.36**

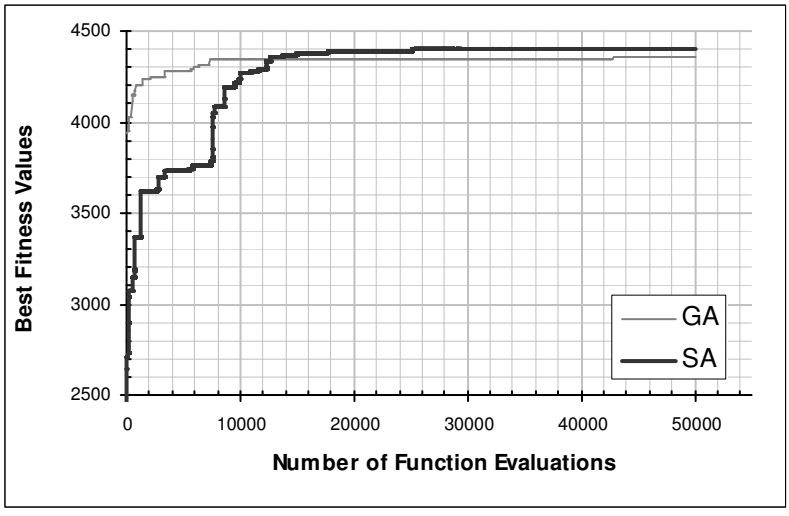


Figure 5.20 Best optimal result found by GA and SA

CHAPTER 6

CONCLUSIONS

A combined simulation-optimization of a coastal aquifer is studied to optimize the maximum benefit by extracting water without seawater intrusion. The simulation formulation was based on the single potential formulation by Strack (1976). The main assumptions were sharp interface between seawater and freshwater and seawater is assumed stagnant while freshwater flows horizontally over seawater. Seawater intrusion is tracked by the potential values at the extraction wells. MODFLOW is used for the numerical solution of the single potential solution. Two different heuristic algorithms, GA and SA are used as optimization tools. Different prevention methods are added to the model and optimal solution for these scenarios are found by using the management formulation. A method named "Alternating Constraints Method" is introduced to improve the results by GA, where locations are also decision variables. To evaluate the effectiveness of the system, linear or mixed integer programming is utilized, whenever applicable. The conclusions are summarized below;

1. Although an optimal parameter set is found and used for solutions with GA, good results can be achieved by using different combinations of parameter values and different type of GA operators.

2. Binary length of decision variables sets the precision of the solution. Longer the string length, larger the search space and it may be harder for the optimization solver to find the near global solution. Thus, before starting a study, the user should decide on the degree of precision important for the solution. This kind of control on precision enables the user to make a coarser search at the beginning of a research without much computational requirement and increase the precision of the solution as research progress.

3. The results by LP and MIP, whenever applicable, showed that GA and SA are successfully finding near global optimums for the seawater management problem considered in this study.

4. Including the locations of recharge elements (i.e. injection wells, canal) as decision variable in the solutions improved the maximum extraction amount. For both injection wells and canal, optimum results are found when they are located farther away from the coast.

5. Alternating Constraints Method improved the results where locations of the recharge elements are of important in terms of management problem. The method improved the results by both increasing the quality of the final solution and decreasing the number of function evaluations required to reach that solution.

6. Adding canal or injection well to the model prevented the encroachment of seawater interface and increased the maximum extraction amount.

7. When injected and extracted water is of the same economical value, there is no net gain in groundwater extraction amount. Using lower quality of injected water increases the efficiency of the groundwater extraction.

8. Maximum benefit achieved using either canal or injection well is almost identical for the management objectives considered. A choice between two options requires other criterias involved in solution (i.e. the cost of installation, sustainability questions or environmental effects)

9. Various runs with different starting points and seed numbers showed that SA usually converges to near global optimum, independent of the starting point or random processes involved in solution by SA.

10. The results show that both GA and SA can effectively be used to obtain near global solutions for the management problems utilized in this study. Computational time depends on the number of decision variables, discretization resolution, and complexity of the problem (i.e. complex geometry).

REFERENCES

Ahlfeld, D. P. and Heidari, M., *Applications of Optimal Hydraulic Control to Ground-Water Systems,* Journal of Water Resources Planning and Management, ASCE, 120(3), 350-365, 1994.

Aly, A. H. and Peralta, R. C., *Comparison of a Genetic Algorithm and Mathematical Programming to the Design of Groundwater Cleanup Systems*, Water Resources Research, 35(8), 2415–2426, 1999.

Back, T., *Optimal Mutation Rates in Genetic Search*, In Forrest, S. (Ed.), Proceedings of the 5th International Conference on Genetic Algorithms, San Mateo, CA: Morgan Kaufmann, 2–9, 1993.

Back, T. and Schutz, M, *Intelligent Mutation Rate Control in Canonical Genetic Algorithms*, In Ras, Z. W. and Michalewicz, M. (Eds.), Proceedings of the 9th International Symposium on Foundations of Intelligent Systems, Berlin: Springer, 1079, 158-167, 1996.

Banzhaf, W. and Reeves, C., *Foundations of Genetic Algorithm 5*, Morgan Kaufmann Publishers, San Francisco, 1999.

Bear, J., Cheng, A. H. D., Sorek, S., Ouazar, D. and Herrera, I. (Eds.), *Seawater Intrusion in Coastal Aquifers - Concepts, Methods and Practices, in Theory and Application of Transport in Porous Media*, Kluwer Academic Publishers, Dordrecht, 1999.

Bear, J. and Zhou, *Seawater Intrusion into Coastal Aquifers*, Chapter 12 in: Delleur, J. W. (Ed.), The Handbook of Groundwater Engineering, Second Edition, CRC Press, New York, 12.1-12.29, 2004.

Benhachmi, M. K., Ouazar, D., Naji, A., Cheng, A.H.-D., and El Harrouni, K., *Optimal Management in Saltwater-Intruded Coastal Aquifers by Simple Genetic Algorithm*, In Ouazar, D. and Cheng, A.H.-D. (Eds.), 1st International Conference on Saltwater Intrusion and Coastal Aquifers-Monitoring, Modeling, and Management, Essaouira, Morocco, April 23-25, 2001.

Benhachmi, M. K., Ouazar, D., Naji, A., Cheng, A.H.-D., and El Harrouni, K., *Pumping Optimization in Saltwater-intruded Aquifers by Simple Genetic Algorithm-Deterministic Model*, Proceedings of Coastal Aquifers Intrusion Technology: Mediterranean Countries International Conference (TIAC'03), Alicante, Spain, 1, 291-293, 2003.

Bhattacharjya, R. K. and Datta, B., *Optimal Management of Coastal Aquifers Using Linked Simulation Optimization Approach*, Water Resources Management, 19(3), 295-320, 2005.

Bohachevsky, I., Johnson, M. E., and Stein, M. L., *Generalized Simulated Annealing for Function Optimization*, Technometrics, 28, 209-217, 1986.

Camur, M. Z. and Yazıcıgil, H., *Effects of the Planned Ephesus Recreational Canal on Freshwater-Seawater*, Environmental Geology, 48(2), 229-237, 2005.

Cheng A. H.–D. and Ouazar D., *Analytical Solutions*, Chapter 6 in: Bear, J. ,Cheng , A. H. D., Sorek, S., Ouazar, D., and, Herrera, I. (Eds.), Seawater Intrusion in Coastal Aquifers, Kluwer Academic Publishers, Dordrecht, 163-191, 1999.

Cheng, A. H.–D., Halhal, D., Naji, A., and Ouazar, D., *Pumping Optimization in Saltwater Intruded Coastal Aquifers*, Water Resources Research, 36(8), 2155-2165, 2000

Coley, D., *An Introduction to Genetic Algorithms for Scientists and Engineers*, World Scientific, London, 1999.

Corona, A., Marchesi, M., Marteni, C., and Ridella, S., *Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm*, ACM Trans. Mathematical Software, 13 (3), 262–280, 1987.

Cunha, M. D. C., *Four Approaches for Groundwater Planning*, Engineering Optimization, 35(1), 39-49 (11), 2003.

Das, A. and Datta, B., *Development of Multi Objective Management Models for Coastal Aquifers*, Journal of Water Resources Planning Management, ASCE, 125(2), 76–87, 1999.

Das, A. and Datta, B., *Simulation Of Seawater Intrusion In Coastal Aquifers: Some Typical Responses*, SADHANA – Academy Proceedings in Engineering Sciences, Indian Academy of Sciences, 26(4), 317-353, 2001.

Davis, L. and Steenstrup, M., *Genetic Algorithms and Simulated Annealing: An Overview*, In Davis, L. (Ed.), Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, Inc., LA, 1-11, 1987.

De Jong, K. A., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

De Jong, K. A., *Genetic Algorithms Are NOT Function Optimizers,* In Whitley, D. (Ed.), Foundations of Genetic Algorithms: Proceedings 24-29 July 1992, Morgan Kaufman, Vail, CO, 1992.

De Jong, K. A., *Evolutionary Computation: A Unified Approach*, MIT Press, 2006.

Delleur, J. W. (Ed.), *The Handbook of Groundwater Engineering*, Second Edition, CRC Press, New York, 2004a.

Delleur, J. W., *Elementary Groundwater Flow and Transport Processes*, Chapter 3 in: Delleur, J. W. (Ed.), The Handbook of Groundwater Engineering, Second Edition, CRC Press, New York, 3.1-3.45, 2004b.

Demirbaş, K., *Combined Optimization-simulation of an Excavation Site for Dewatering Purposes*, MS Thesis, Middle East Technical University, Civil Eng. Dept., Ankara, 2003.

Emch, P. G. and Yeh, W. W. G., *Management Model for Conjunctive Use of Coastal Surface Water and Groundwater*, Journal of Water Resources Planning and Management, ASCE, 124(3), 129-139, 1998.

Essaid, H. I., *A Comparison of The Coupled Fresh Water-Salt Water Flow and The Ghyben-Herzberg Sharp Interface Approaches to Modeling of Transient Behavior in Coastal Aquifer Systems*, Journal of Hydrology, 169-193, 1986.

Essaid, H. I., *A Multilayered Sharp Interface Model of Coupled Freshwater and Saltwater Flow in Coastal Systems: Model Development and Application*, Water Resources Research, 26 (7), 1431-1454., 1990a.

Essaid, H. I., *The Computer Model, SHARP, A Quasi-Three-Dimensional Finite-Difference Model to Simulate Freshwater and Saltwater Flow in Layered Coastal Aquifer Systems: U.S. Geological Survey Water-Resources Investigations Report 90-4130*, 1990b.

Essaid, H. I., *USGS SHARP Model*, Chapter 8 in: Bear, J. ,Cheng , A. H. D., Sorek, S., Ouazar, D., and, Herrera, I. (Eds.), Seawater Intrusion in Coastal Aquifers, Kluwer Academic Publishers, Dordrecht, 213-247, 1999.

Finney, B. A., Samsuhadi, S., and Willis, R., *Quasi-three-dimensional Optimisation Model for Jakarta Basin*, Journal of Water Resources Planning and Management, 118, 18–31, 1992.

Galeati, G., Gambolati, G., and Neuman, S.P., *Coupled and Partially Coupled Eulerian-Lagrangian Model of Freshwater-Saltwater Mixing*, Water Resources Research, 28(1), 149-165, 1992.

Goffe, W. L., Ferrier, G. D., and Rodgers, J., *Simulated Annealing: An Initial Application in Econometrics*, Computer Science in Economics & Management, Springer, 5(2), 133-146, 1992.

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

Gorelick, S. M., *A Review of Distributed Parameter Management Modeling Methods*, Water Resources Research, 19(2), 305-319, 1983.

Greenwald, R., *MODMAN: An Optimization Module for MODFLOW Version 4.0*, GeoTrans, 2 Paragon Way, Freehold, NJ 07728., 1998.

Grefenstette, J., *Optimization of Control Parameters for Genetic Algorithms*, IEEE Transactions on Systems, Man, and Cybernetics, SMC-16(1), 122–128, 1986.

Guo, W. and Langevin, C. D., *User's guide to SEAWAT: A Computer Program for Simulation of Three-Dimensional Variable-Density Ground-Water Flow*, U.S. Geological Survey Open-File Report 01-434, 2002.

Hallaji, K. and Yazıcıgil, H., *Optimal Management of a Coastal Aquifer in Southern Turkey*, Journal of Water Resources Planning and Management, ASCE, 122, 233–244, 1996.

Harbaugh, A. W. and McDonald, M. G., *User's Documentation for MODFLOW-96, an Update to the U.S. Geological Survey Modular Finite-Difference Ground-Water Flow Model: U.S. Geological Survey Open-File Report 96-485*, 1996.

Harbaugh, A. W., Banta, E. R., Hill, M. C., and McDonald, M. G., *MODFLOW-2000, the U.S. Geological Survey Modular Ground-Water Model - User Guide to Modularization Concepts and the Ground-water Flow Process: U.S. Geological Survey Open-File Report 00-92*, 2000

Haupt, R. L. and Haupt, S. E., *Practical Genetic Algorithms*, Wiley, New York, 2004.

Henry, H. R., *Salt Intrusion into Freshwater Aquifers*, Journal of Geophysical Research, 64, 1911-1919, 1959.

Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

Huyakorn, P. S., Wu, Y. S., and Park, N. S., *Multiphase Approach to the Numerical Solution of a Sharp Interface Saltwater Intrusion Problem*, Water Resources Research, 32(1), 93–102, 1996.

Karahanoğlu, N. and Doyuran, V., *Finite Element Simulation of Seawater Intrusion into a Quary-Site Coastal Aquifer*, Kocaeli Darıca, Turkey, Environmental Geology, 44(4), 456-466, 2003.

Katsifarakis, K. L. and Petala, Z., *Combined Combined Use of Genetic Algorithms and Boundary Elements to Optimize Coastal Aquifers Protection and Restoration of Environment VII*, Mykonos, 2004.

Katsifarakis, K. L. and Petala, Z., *Combining Genetic Algorithms and Boundary Elements to Optimize Coastal Aquifers' Management*, Journal of Hydrology, 327(1-2), 200-207, 2006.

Kirkpatrick, S., Gelatt Jr., C. D., and Vecchi, M. P., *Optimization by Simulated Annealing*, Science, 220, Number 4598, 671-680, 1983.

Kuo, C. H., Michel, A. N., and Gray, W. G., *Design of Optimal Pump-and-Treat Strategies For Contaminated Groundwater Remediation Using the Simulated Annealing Algorithm*, Water Resources Research ,15(2), 95-105, 1992.

Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

Leap, D., *Geological Occurrence of Groundwater*, Chapter 2 in: Delleur, J. W. (Ed.), The Handbook of Groundwater Engineering, Second Edition, CRC Press, New York, 2.1-2.59, 2004.

Lin, Y. C. and Yeh, H.D., *Identifying Groundwater Pumping Source Information Using Simulated Annealing*, Published online in Wiley Interscience, 2008.

Lindo Systems, *LINDO User's Manual*, Lindo Systems, Chicago, IL, 1996.

Mantoglou, A., *Pumping Management of Coastal Aquifers Using Analytical Models of Saltwater Intrusion*, Water Resources Research, 39(12), 1-12, 2003.

Mantoglou, A., Papantoniou, M., and Giannoulopoulos, P., *Management of Coastal Aquifers Based on Nonlinear Optimization and Evolutionary Algorithms*, Journal of Hydrology, 297(1-4), 209-228., 2004.

Mantoglou, A. and Papantoniou, M., *Optimal Design of Pumping Networks in Coastal Aquifers Using evolutionary algorithms, Proceedings of International Symposium on Water Resources Management: Risk and Challenges for the 21st Century*, İzmir, 783-793. 2004.

Mantoglou, A. and Papantoniou, M., *Optimal Design of Pumping Networks in Coastal Aquifers Using Sharp Interface Models*, Journal of Hydrology, 361 (52-63), 2008.

McDonald, M. G. and Harbaugh, A. W., *A Modular Three-Dimensional Finite Difference Groundwater Flow Model*, Modelling Techniques, Book 6, Chapter A1, 1988.

McKinney, D. C. and Lin, M.-D., *Genetic Algorithms Solution of Groundwater Management Models*, Water Resources Research, 30(6), 1897-1906, 1994.

Mercer J. W., Larson, S. P., and Faust, C. R., *Simulation of Salt Water Interface Motion, Ground Water*, 18, 374-385, 1980.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, M. N., and Teller, E., *Equations of State Calculations By Fast Computing Machines*, Journal of Chemical Physics, 21, 1087-1092, 1953.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs (3rd ed.), Springer-Verlag,* London, 1996.

Motz, L. H., Yurtal, R., and Gordu F., *Final Project Report: Optimization of Groundwater Use Subject to Saltwater Intrusion Along the Mediterranean Cost of Turkey for National Science Foundation and Turkish Scientific and Technical Council of Turkey*, 2004.

Murtagh, B. A. and Saunders, M. A., *MINOS 5.4 User's Guide. Technical Report*, SOL 83–20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California, 1993.

Qahman, K. and Larabi, A., *Evaluation and Numerical Modeling of Seawater Intrusion in the Gaza Aquifer (Palestine)*, Hydrogeology Journal, 2005.

Papadimitriou, C. H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Park, C. H. and Aral, M. M., *Multi-Objective Optimization of Pumping Rates and Well Placement in Coastal Aquifers*, Journal of Hydrology, 290 (1-2), 80-99, 2004.

Park, N., Hong, S. H., Shim, M. G., Han, S. Y., and Bae, S. K., *Optimization of Ground Water Withdrawal in Coastal Regions. Second International Conference on Saltwater Intrusion and Coastal Aquifers' Monitoring, Modeling and Management (SWICA-M3)*, Merida, Yucatan, Mexico, March 30-April 2, 2003.

Polo, J. F. and Ramis, F. J. R., *Simulation of Salt Water-Fresh Water Interface Motion*, Water Resources Research, 19, 61-68, 1983.

Ranjan, P., Kazama, P. and Sawamoto, M., *Modeling of the Dynamics of Saltwater-Freshwater, Interface in Coastal Aquifers, Proceedings of the Joint AOGS Annual Meeting & APHW Second Conference 2004*, Singapore, 373-380, July 2004.

Rao, S. V. N., Thandaveswara, B. S., Murty Bhallamudi, S., and Srinivasulu, V., *Optimal Groundwater Management in Deltaic Regions Using Simulated Annealing and Neural Networks*, Water Resources Management, 17, 409-428 , 2003.

Rao, S. V. N., Sreenivasulu, V., Bhallamudi, S.M., Thandaveswara, B.S., and Sudheer, K.P. *Planning Groundwater Development in Coastal Aquifers*, Hydrological Sciences Journal 49(1), 155-170, 2004.

Sakr, S. A., *Validity of a Sharp Interface Model in a Confined Coastal Aquifer*, Hydrogeology Journal, 7(2), 155-160, 1999.

Schaffer, J. D., Caruana, R. A., Eshelman, L. J., and Das, R., *A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization*, Proceedings of 3rd International Conference on Genetic Algorithms, San Mateo, CA: Morgan Kaufmann, 51–60, 1989.

Shamir, U. and Dagan, G., *Motion of the Seawater Interface in Coastal Aquifers: A Numerical Solution*, Water Resources Research, 7, 644-657, 1971.

Shamir, U., Bear, J., and Gamliel, A., *Optimal Annual Operation of A Coastal Aquifer*, Water Resources Research, 20, 435–444, 1984.

Sivanandam, S. N. and Deepa, S. N., *Introduction to Genetic Algorithms.* Springer-Verlag, 2007.

Strack, O. D. L., *A Single-potential Solution for Regional Interface Problems in Coastal Aquifers*, Water Resources Research, 12(6), 1165-1174, 1976.

Todd, D. K., , *Ground-water Hydrology (Second Edition):* John Wiley and Sons, New. York, 1980.

Tokgöz, M., Yılmaz, K. K., and Yazıcıgil, H., *Optimal Aquifer Dewatering Schemes For Excavation of a Collector Line*, Journal of Water Resources Planning and Management, ASCE, 248-261, 2002.

Van Dam, J. C., *Exploitation, Restoration and Management*, Chapter 4 in: Bear, J. ,Cheng , A. H. D., Sorek, S., Ouazar, D., and, Herrera, I. (Eds.), Seawater Intrusion in Coastal Aquifers, Kluwer Academic Publishers, Dordrecht, 73-125, 1999.

Van Laarhoven, P. J. M. and Aarts, E. H. L., *Simulated Annealing, Theory and Practice*, Kluwer Acad. Publ., 1987.

Vanderbilt, D. and Louie, S.G., *A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables*, J. Comput. Phys., 56, 259-271, 1984.

Voss, C. I., *SUTRA (Saturated Unsaturated Transport): A Finite-element Simulation Model for Saturated-unsaturated, Fluid-density-dependent Ground-water Flow with Energy Transport or Chemically-Reactive Single-species Solute Transport*, U.S. Geological Survey Water-Resources Investigations Report p. 84-4369, 1984.

Voss, C. I. and Souza W.R., *Variable Density Flow and Solute Transport Simulation of Regional Aquifers Containing a Narrow Freshwater-Saltwater Transition Zone*, Water Resources Research, 23(10), 1851-1866, 1987.

Wang, M. and Zheng, Y., *Ground Water Management Optimization Using Genetic Algorithms And Simulated Annealing: Formulation And Comparison*, Journal of the American Water Resources Association, 34(3), 519–530, 1998.

Willis, R. and Liu, P., *Optimisation Model for Groundwater Planning,* Journal of Water Resources Planning and Management, ASCE, 110, 333-347, 1984.

Willis R. and Finney, B. A., *Planning Model for Optimal Control of Saltwater Intrusion*, Journal of Water Resources Planning and Management, ASCE, 114, 163–178, 1988.

Wilson, J. L. and Costa, S. A., *Finite Element Simulation of a Saltwater/Freshwater Interface With Indirect Toe Tracking*, Water Resources Research, 18(4), 1069-1080, 1982.

Zhou, X., Chen, M., and Liang, C., *Optimal Schemes of Groundwater Exploitation for Prevention of Seawater Intrusion in the Leizhou Peninsula in Southern China*, Environmental Geology, 43, 978-985, 2003.

APPENDIX A

GA CODE

Genkod.f90

```
Program GENETIK_INJLOCX
INTEGER seed
INTEGER pop_num,sel_int1,sel_int2,gen_num,xd,eni_sira,enk_sira
REAL sel_num1,sel_num2,fitness(1000),fitness_new(1000),fitness_sum(1000)
REAL inj_coef
REAL fitness_eni,fitness_ort,totalx,bestever_fitness,fitness_ortiki
INTEGER str_length(100),uni_xover(100),x_overpt,x_overpt2,ix,jx,eni_gen,gen,mut_adet,x_num
INTEGER p,totstr_length,runno,top_stra,top_strb,xover_type,mut_type,sca_type,sel_type
REAL up_bound(30),low_bound(30),bi_constant(30),rastgele !Max no. of decision variables is 30
REAL toplam(1000),deneme,x(1000,30),best_x(30),worst_x(30),bestever_x(30)
REAL mut_num,f,con,pen_con, r,rnew,rnew2,rara,zz,host,best_all(1000,1000),avg_all(1000,1000)
REAL var_all(1000,1000),a_constant,b_constant,sca_constant,gen_var
INTEGER time,bas_time,bit_time,tot_time,inp_say,totinp_say,penalty_coef,elitism
REAL Q(50) ! Max. number of wells is 50.
INTEGER birey(1000,100),birey_ara(1000,100),run_say,best_birey(100),bestever_birey(100)
INTEGER sel_intara1,sel_intara2,pop_transfer,tot_saat,tot_dak,tot_san,total_run
INTEGER seed_type,numfit_zero
CHARACTER*3 :: cg
CHARACTER*5 :: ch

seed=99987687 ! Random seed number.
pop_transfer=0 ! If 0 no population transfer, if one for once, brings the last population to next run.
seed_type=1    ! If 0 seed remains same, else it changes at each input cycle.
total_run=60   ! If run number is greater than input file no. hen cycle in input files with different seed.
totinp_say=3    ! Number of input files to be used.

DO 100 run_say=1,total_run   !N o of input txt file
bas_time=time()
OPEN(unit=17,file='c:\Korkut_GA\\RUNNO.txt')
READ (17,*) RUNNO
CLOSE (17)
OPEN(unit=18,file='c:\Korkut_GA\\RUNNO.txt',status='replace')
WRITE (18,*) RUNNO+1
CLOSE(18)

WRITE (ch,2000) RUNNO
2000  FORMAT(I5)

! Selects the input file according to run_say.
IF (MOD(run_say,totinp_say).EQ.0) THEN
inp_say=totinp_say
ELSE
inp_say=MOD(run_say,totinp_say)
END IF
WRITE (cg,1000) inp_say
1000  FORMAT(I3)

! Read Input files.
OPEN(5,file='input'//cg//'.txt')
IF (seed_type.EQ.0) THEN
```

```
seed=seed
ELSE
IF (totinp_say.EQ.1) THEN
seed=seed-100
ELSE IF (MOD(run_say,totinp_say).EQ.1) THEN
seed=seed-100
ELSE
seed=seed
END IF
END IF

READ (5,*) sel_type ! If 0 random, 1 fitness proportianete, if 2 tournament selection
READ (5,*) xover_type ! If 1 one point, if 2 two point, if three uniform xover
READ (5,*) mut_type ! If 1 normal muation, if 2 uniform mutation
READ (5,*) sca_type ! If 0 it is off, else on
READ (5,*) sca_constant !will be included in calculations if scaling is on
READ (5,*) elitism ! If 0 it is off, else on
READ (5,*) x_num
DO k1=1,x_num
READ (5,*) low_bound(k1)
READ (5,*) up_bound(k1)
ENDDO
DO k1=1,x_num
READ (5,*) str_length(k1)
ENDDO
READ (5,*) pop_num
READ (5,*) gen_num
READ (5,*) x_over
READ (5,*) mut_num
READ (5,*) penalty_coef
READ (5,*) inj_coef
CLOSE(5)

eni=0
totstr_length=0
enk_sira=1
eni_sira=1
bestever_fitness=0 ! best ever fitness

DO k2=1,x_num
bi_constant(k2)=(up_bound(k2)-low_bound(k2))/(2**str_length(k2)-1)
totstr_length=str_length(k2)+totstr_length
ENDDO
RESULT = MAKEDIRQQ('sonuclar\sonuclar'//ch//'')
OPEN(10,file='sonuclar\\sonuclar'//ch//'\output.txt')
WRITE(10,*) 'PROGRAM CIKTILARI'
OPEN(20,file='sonuclar\\sonuclar'//ch//'\kontrol.txt')
WRITE(20,*) 'KONTROL CIKTILARI'
OPEN(30,file='sonuclar\\sonuclar'//ch//'\xover_kontrol.txt')
OPEN(40,file='sonuclar\\sonuclar'//ch//'\ortalama_fitness.txt')
OPEN(50,file='sonuclar\\sonuclar'//ch//'\mutas_kontrol.txt')
OPEN(70,file='sonuclar\\sonuclar'//ch//'\penaltilar.txt')
OPEN(80,file='sonuclar\\sonuclar'//ch//'\best_fitness.txt')
OPEN(90,file='sonuclar\\sonuclar'//ch//'\gen_variation.txt')

WRITE (10,*) 'Total Run Number:',run_say,'      /',total_run
WRITE (10,*) 'Degisken Sayısı:',x_num
DO k1=1,x_num
WRITE (10,*) 'Degisken',(k1),'in aralığı:',low_bound(k1),up_bound(k1)
ENDDO
DO k1=1,x_num
WRITE (10,*) 'Degisken',(k1),'in string uzunluğu:',str_length(k1)
```

```
ENDDO
WRITE (10,*) 'Populasyon Sayısı:',pop_num
WRITE (10,*) 'Generasyon Sayısı:',gen_num
WRITE (10,*) 'Crossover Katsayısı:',x_over
WRITE (10,*) 'Mutasyon Katsayısı:',mut_num
WRITE (10,*)
WRITE (10,15) ' Selection türü:',sel_type,'(if 0 random sel, 1 fitness proportianete, if tournament)'
WRITE (10,15) ' Crossover türü:',xover_type,'(if 1 one point, if 2 two point, if three uniform)'
WRITE (10,15) ' Mutasyon türü:',mut_type,'(if 1 normal muation, if 2 asamali mutation)'
WRITE (10,15) ' Scaling durumu off/on:',sca_type,'(if 0 it is off else on)'
WRITE (10,15) ' Elitism durumu off/on:',elitism,'(if 0 it is off 1 on)'
WRITE (10,15) ' Populasyon transferi off/on:',pop_transfer,'   (if 0 it is off else on)'
WRITE (10,15) ' Degisik Seed off/on:',seed_type,'   (if 0 it is off else on)'
WRITE (10,*) ' Input sayısı:',totinp_say
WRITE (10,*) ' Penalti Katsayisi:',penalty_coef
WRITE (10,*) ' Enjeksiyon Kaysayisi:',inj_coef
WRITE (10,*) ' Seed sayısı:',seed
WRITE (10,*) ' Scaling katsayısı:',sca_constant
WRITE (10,*)
WRITE (80,*) 'Runno:',ch
WRITE (80,*) pop_num
WRITE (80,*) gen_num
WRITE (80,*) x_over
WRITE (80,*) mut_num
WRITE (80,*) sel_type
WRITE (80,*) xover_type
WRITE (80,*) mut_type
WRITE (80,*) penalty_coef
WRITE (80,*) seed
WRITE (80,*)
WRITE (40,*) 'Runno:',ch
WRITE (40,*) pop_num
WRITE (40,*) gen_num
WRITE (40,*) x_over
WRITE (40,*) mut_num
WRITE (40,*) sel_type
WRITE (40,*) xover_type
WRITE (40,*) mut_type
WRITE (80,*) penalty_coef
WRITE (40,*) seed
WRITE (40,*)
15 FORMAT (a21,i3,a56)

CALL SRAND(seed)

IF (pop_transfer.EQ.0) THEN
DO i=1,pop_num,1
DO j=1,totstr_length,1
IF(rand().GE.0.5) then
birey(i,j)=1.
ELSE
birey(i,j)=0.
END IF
END DO
END DO

ELSE IF (pop_transfer.EQ.1) THEN
IF (MOD(run_say,2).EQ.1) THEN
birey=0 !Reset individuals.
toplam=0
x=0
```

```
! Random individuals are built.
DO i=1,pop_num,1
DO j=1,totstr_length,1
IF(RAND().GE.0.5) then
birey(i,j)=1.
ELSE
birey(i,j)=0.
END IF
END DO
END DO

ELSE
!Last pop. of the previous pop is transformed into current pop if
! pop transfer is selected.

DO p=1,pop_num
top_stra=1
top_strb=0
DO xd=1,x_num
toplam(xd)=(x(p,xd)-low_bound(xd))/bi_constant(xd)
top_strb=top_strb+str_length(xd)
DO j=top_stra,top_strb
IF (toplam(xd).GE.2**(top_strb-j)) then
birey(p,j)=1
toplam(xd)=toplam(xd)-2**(top_strb-j)
ELSE
birey(p,j)=0
END IF
END DO
top_stra=top_stra+str_length(xd)
END DO
END DO
END IF
END IF

DO 200 gen=1,gen_num,1  ! Generation loop

WRITE (10,*)
WRITE (20,*)
WRITE (30,*)
WRITE (50,*)
WRITE (70,*)
WRITE (10,*) 'Generasyon Sayısı=',gen
WRITE (20,*) 'Generasyon Sayısı=',gen
WRITE (30,*) 'Generasyon Sayısı=',gen
WRITE (50,*) 'Generasyon Sayısı:',gen
WRITE (70,*) 'Generasyon Sayısı:',gen

! Evaluate the fenotype of individuals.
DO 600 p=1,pop_num,1

top_stra=1
top_strb=0
DO xd=1,x_num
toplam(xd)=0
top_strb=top_strb+str_length(xd)
DO j=top_stra,top_strb   ! Sum of the bit values for indvs.
toplam(xd)=toplam(xd)+birey(p,j)*2**(top_strb-j)
END DO
x(p,xd)=bi_constant(xd)*toplam(xd)+low_bound(xd)
top_stra=top_stra+str_length(xd)
END DO
```

```
600 END DO

WRITE (10,13) 'Pop_num',('x[o]',i=1,x_num),'TotalExtQ','TotalNetQ','Fitness','Penalty'
13 FORMAT (a13,30a12)

fitness_eni=0 ! Best individual in current pop
fitness_enk=10000000 ! Worst individual in current pop
numfit_zero=0   ! Number of indv. with zero fitness

DO 500 p=1,pop_num

! Calls the "fonksiyon" subroutine.
f = FUNC(x,p,x_num,ch,inj_coef,run_say,penalty_coef)
fitness(p)=f

IF (fitness(p).GT.fitness_eni) THEN ! Finds the best fitnes in current pop.
fitness_eni = fitness(p)
DO j=1,x_num
best_x(j)=x(p,j)
ENDDO
DO J=1,totstr_length
best_birey(j)=birey(p,j)
END DO
END IF

IF (fitness(p).LT.fitness_enk) THEN ! Finds the worst indv.in current pop.
fitness_enk = fitness(p)
enk_sira=p
END IF

IF (fitness_eni.GT.bestever_fitness) then
DO j=1,x_num
bestever_x(j)=best_x(j)
ENDDO
DO J=1,totstr_length
bestever_birey(j)=best_birey(j)
END DO
bestever_fitness=fitness_eni
eni_gen=gen
eni_sira=p
END IF
500 END DO

!!!!!!!!!!!!!!!!!!!!!!!
! Elitist !!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!
IF (elitism.EQ.1) THEN
IF (gen.GT.1) THEN

DO j=1,totstr_length
birey(enk_sira,j)=bestever_birey(j)
END DO

DO j=1,x_num
x(enk_sira,j)=bestever_x(j)
ENDDO
fitness(enk_sira)=bestever_fitness

WRITE (10,18) 'Fittest birey,',enk_sira,'.nolu yukarıdaki en az fit bireyin yerini alır'
WRITE (10,8) enk_sira,'.',(x(enk_sira,I),I=1,x_num),fitness(enk_sira),gen
18 FORMAT (a13,i5,a50)
```

```
8 FORMAT (i,a,20f10.2)
END IF

ELSE
END IF
!!!!!!!!!!!!!!!!!!!!!!!!!

! Negative fitness takes the 0 value
DO p=1,pop_num,1
IF (fitness(p).LE.0.) then ! Fitness' should be positive
fitness(p)=0.
numfit_zero=numfit_zero+1  ! Number of indv. with zero fitness
ELSE
ENDIF
END DO

! Sum up the fitness values for fitness proportinate selection
fitness_sum(1)=fitness(1)
DO p=1,pop_num-1,1
fitness_sum(p+1)=fitness_sum(p)+fitness(p+1)
END DO
fitness_ort=fitness_sum(pop_num)/pop_num

WRITE (10,19) 'En iyi fitness=',bestever_fitness,eni_gen,'. generasyondan',eni_sira,'. no.daki birey'
19 FORMAT (a18,f9.2,i5,a15,i5,a15)
DO j=1,x_num
WRITE (10,*) 'x degerleri=',bestever_x(j)
END DO

WRITE (40,*) fitness_ort
WRITE (80,*) bestever_fitness
avg_all(gen,run_say)=fitness_ort
best_all(gen,run_say)=bestever_fitness

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Linear Scaling !!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
IF (sca_type.EQ.1) THEN
! Zero fitness individuals are not included for scaling calc.
fitness_ort=fitness_sum(pop_num)/(pop_num-numfit_zero)
fitness_sum(1)=0
a_constant=(sca_constant-1)*fitness_ort/(fitness_eni-fitness_ort)
b_constant=(1-a_constant)*fitness_ort

DO p=1,pop_num
fitness(p)=a_constant*fitness(p)+b_constant
IF (fitness(p).LE.0) THEN
fitness(p)=0
END IF
WRITE (10,8) p,'.yeni fitness',fitness(p)
END DO

DO p=1,pop_num-1,1
fitness_sum(p+1)=fitness_sum(p)+fitness(p+1)
END DO

WRITE (10,*) 'Avarage fitness for scaling',fitness_ort
END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Genetic Variation !!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
k=0
DO i=1,pop_num
DO j=1,totstr_length
IF (birey(i,j).EQ.best_birey(j)) THEN
k=k+1
ENDIF
END DO
END DO

gen_var=k/(totstr_length*(pop_num*1.))
WRITE (90,*) gen_var
var_all(gen,run_say)=gen_var
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

DO 300 p=1,pop_num/2
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! SELECTION !!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Random selection !!!!!!!!!!!!!!

IF (sel_type.EQ.0) THEN

! First Individual
sel_num1=RAND()*pop_num
sel_int1=sel_num1+1

! Second Individual
sel_num2=RAND()*pop_num
sel_int2=sel_num2+1

! Fitness proportinate selection !!!!!

ELSE IF (sel_type.eq.1) THEN
sel_num1=rand()*fitness_sum(pop_num)
sel_num2=rand()*fitness_sum(pop_num)

! Select first individual
k=1
DO WHILE(sel_num1.GE.fitness_sum(k))
k=k+1
ENDDO
sel_int1=k

! Select second individual
k=1
DO WHILE(sel_num2.GE.fitness_sum(k))
k=k+1
ENDDO
sel_int2=k

!Tournament selection !!!!!!!!!!!

ELSE IF (sel_type.EQ.2) then
 ! First Individual
sel_num1=RAND()*pop_num
sel_num2=RAND()*pop_num
sel_intara1=sel_num1+1
sel_intara2=sel_num2+1

IF (fitness(sel_intara1).GT.fitness(sel_intara2)) THEN
sel_int1=sel_intara1
```

```
ELSE
sel_int1=sel_intara2
END IF

WRITE(20,*) 'Çift Selection.',p
WRITE(20,9) 'Tournamenta secilenler',sel_intara1,'. ve',sel_intara2,'.birey'
WRITE(20,*) 'Fitnesları',fitness(sel_intara1),'. ve',fitness(sel_intara2)
WRITE(20,*) 'Crossovera hak kazanan birinci birey',sel_int1

! Second Individual
sel_num1=RAND()*pop_num
sel_num2=RAND()*pop_num
sel_intara1=sel_num1+1
sel_intara2=sel_num2+1

IF (fitness(sel_intara1).GT.fitness(sel_intara2)) THEN
sel_int2=sel_intara1
ELSE
sel_int2=sel_intara2
END IF

WRITE(20,9) 'Tournamenta secilenler',sel_intara1,'. ve',sel_intara2,'.birey'
WRITE(20,*) 'Fitnesları',fitness(sel_intara1),'. ve',fitness(sel_intara2)
WRITE(20,*) 'Crossovera hak kazanan ikinci birey',sel_int2

END IF

! SELECTION ENDS !!!!!!!!!!!!!!!!!!!!!!!!!!!!

WRITE(20,9) 'Selectiona secilen',sel_int1,'. ve',sel_int2,'.bireyler'
9 FORMAT (a,i,a,i,a)

WRITE(30,*) 'Crossover oncesi'
WRITE(30,10 ) (birey(sel_int1,y),y=1,totstr_length)
WRITE(30,10 ) (birey(sel_int2,y),y=1,totstr_length)
10 FORMAT (100i2)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! XOVER !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! One point crossover !!!!!!!!!!!

IF (xover_type.EQ.1) THEN
!If random number less than xover constant, then xover
r=Rand()
IF (r.LE.x_over) THEN
rnew=RAND()
x_overpt=(totstr_length-1)*rnew+1. !The point of xover

! Bits remain same till the point of xover.
DO c=1,x_overpt,1
Birey_ara(2*p-1,c)=Birey(sel_int1,c)
Birey_ara(2*p,c)=Birey(sel_int2,c)
ENDDO
! Bits interchanged after the point of xover
DO c=x_overpt+1,totstr_length,1
Birey_ara(2*p-1,c)=Birey(sel_int2,c)
Birey_ara(2*p,c)=Birey(sel_int1,c)
ENDDO

! If no xover, the indvs. remain same
```

```
ELSE
x_overpt=0
DO c=1,totstr_length,1
Birey_ara(2*p-1,c)=Birey(sel_int1,c)
Birey_ara(2*p,c)=Birey(sel_int2,c)
ENDDO
ENDIF

WRITE(30,*) 'Crossover sonrası'
WRITE(30,11) (birey_ara(2*p-1,y),y=1,totstr_length)
WRITE(30,11) (birey_ara(2*p,y),y=1,totstr_length)
WRITE(30,16) 'Crossover degeri=',r,rnew,'One-point xover noktası=',x_overpt
16  FORMAT (a20,f7.3,f7.3,a25,i3)
WRITE(30,*)
! Two point crossover !!!!!!!!

ELSE IF (xover_type.EQ.2) THEN

! If random number less than xover constant, then xover
r=RAND()
IF (r.LE.x_over) THEN

rnew=RAND()
rnew2=RAND()

IF (rnew.GT.rnew2) THEN
rara=rnew2
rnew2=rnew
rnew=rara
END IF

x_overpt=(totstr_length-1)*rnew+1. ! the point of xover
x_overpt2=(totstr_length-1)*rnew2+1. ! the point of xover

! Bits remain same till the first point of xover.
DO c=1,x_overpt,1
Birey_ara(2*p-1,c)=Birey(sel_int1,c)
Birey_ara(2*p,c)=Birey(sel_int2,c)
ENDDO
! Bits between point 1 and 2 remains same.
DO c=x_overpt+1,x_overpt2,1
Birey_ara(2*p-1,c)=Birey(sel_int2,c)
Birey_ara(2*p,c)=Birey(sel_int1,c)
ENDDO
! Bits after 2nd point of xover2 remain same.
DO c=x_overpt2+1,totstr_length
Birey_ara(2*p-1,c)=Birey(sel_int1,c)
Birey_ara(2*p,c)=Birey(sel_int2,c)
ENDDO

! If no xover, the indvs. remain same.
ELSE
x_overpt=0
x_overpt2=0
rnew=0
rnew2=0
DO c=1,totstr_length,1
Birey_ara(2*p-1,c)=Birey(sel_int1,c)
Birey_ara(2*p,c)=Birey(sel_int2,c)
ENDDO
ENDIF
```

```
WRITE(30,*) 'Crossover sonrası'
WRITE(30,11) (birey_ara(2*p-1,y),y=1,totstr_length)
WRITE(30,11) (birey_ara(2*p,y),y=1,totstr_length)
WRITE(30,*) 'Crossover degerleri=',r,rnew,rnew2,'Two-point xover noktaları=',x_overpt,x_overpt2
WRITE(30,*)

! Uniform crossover !!!!!!!!!!!!!!
ELSE IF (xover_type.eq.3) THEN

r=RAND()
IF (r.LE.x_over) THEN

! Uniform xover points are selected.
DO j=1,totstr_length,1
IF(RAND().GE.0.5) THEN
uni_xover(j)=1.
ELSE
uni_xover(j)=0.
END IF
END DO

DO c=1,totstr_length,1

IF (uni_xover(c).EQ.1) THEN !make xover
Birey_ara(2*p-1,c)=Birey(sel_int2,c)
Birey_ara(2*p,c)=Birey(sel_int1,c)

ELSE IF (uni_xover(c).EQ.0) THEN
Birey_ara(2*p-1,c)=Birey(sel_int1,c)
Birey_ara(2*p,c)=Birey(sel_int2,c)

END IF
ENDDO

WRITE(30,*) 'Crossover degeri=',r,'Uniform xover string='
WRITE(30,11) (uni_xover(j),j=1,totstr_length)
WRITE(30,*) 'Crossover sonrası'
WRITE(30,11) (birey_ara(2*p-1,y),y=1,totstr_length)
WRITE(30,11) (birey_ara(2*p,y),y=1,totstr_length)

ELSE

x_overpt=0
DO c=1,totstr_length,1
Birey_ara(2*p-1,c)=Birey(sel_int1,c)
Birey_ara(2*p,c)=Birey(sel_int2,c)
ENDDO

WRITE(30,*) 'Crossover degeri=',r
WRITE(30,*) 'No Crossover'
WRITE(30,*) 'Crossover sonrası'
WRITE(30,11) (birey_ara(2*p-1,y),y=1,totstr_length)
WRITE(30,11) (birey_ara(2*p,y),y=1,totstr_length)
WRITE(30,*)
ENDIF

11 FORMAT (100i2)
ENDIF

!! XOVER ENDS !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

300 END DO
```

```
IF (GEN.LT.GEN_NUM) THEN ! No xover after last generation.
DO i=1,pop_num
DO j=1,totstr_length
birey(i,j)=birey_ara(i,j)
END DO
END DO

IF (MOD(pop_num,2).EQ.1) THEN
!If the no. of populasyon is odd, the last individual is written twice
DO j=1,totstr_length
birey(pop_num,j)=birey(pop_num-1,j)
birey_ara(pop_num,j)=birey(pop_num-1,j)
END DO
END IF

END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! MUTATION !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!Normal Mutation !!!!!!!

IF (mut_type.EQ.1) THEN

DO 400 ix=1,pop_num,1
mut_adet=0
DO jx=1,totstr_length,1

rastgele=RAND()
IF(rastgele.LE.mut_num) THEN
WRITE(50,*) 'Mut.Deg.ve Random Mut.Deg',mut_num,rastgele
WRITE(50,*) 'Mutasyon noktası=',jx

IF (birey(ix,jx).EQ.1) THEN
birey_ara(ix,jx)=0
mut_adet=mut_adet+1
ELSE
birey_ara(ix,jx)=1
mut_adet=mut_adet+1
ENDIF
ENDIF

END DO

WRITE(50,*) 'Mutasyon sayısı=',mut_adet
WRITE(50,*) 'Mutasyon oncesi - Aşağı Ok'
WRITE(50,14 ) (birey(ix,y),y=1,totstr_length)
WRITE(50,14) (birey_ara(ix,y),y=1,totstr_length)
WRITE(50,*) 'Mutasyon sonrası - Yuk Ok'
WRITE(50,*)
14  FORMAT (100i2)

400 END DO

! Uniform Mutasyon !!!!!!!

ELSE IF (mut_type.EQ.2) THEN
DO 700 ix=1,pop_num,1
mut_adet=0
top_stra=1
```

```
top_strb=0

DO xd=1,x_num
top_strb=top_strb+str_length(xd)

DO jx=top_stra,top_strb
rastgele=RAND()

IF (rastgele.LE.mut_num/(2**(top_strb-jx+1))) THEN
WRITE(50,*) 'Aşamalı Mut.Deg. ve Random        Mut.Deg',mut_num*100/(2**(top_strb-jx+1)),rastgele
WRITE(50,*) 'Mutasyon noktası=',jx

IF (birey(ix,jx).EQ.1) THEN
birey_ara(ix,jx)=0
mut_adet=mut_adet+1
ELSE
birey_ara(ix,jx)=1
mut_adet=mut_adet+1
ENDIF
ENDIF
END DO
top_stra=top_stra+str_length(xd)
END DO

WRITE(50,*) 'Mutasyon sayısı=',mut_adet
WRITE(50,*) 'Mutasyon oncesi - Aşağı Ok'
WRITE(50,14) (birey(ix,y),y=1,totstr_length)
WRITE(50,14) (birey_ara(ix,y),y=1,totstr_length)
WRITE(50,*) 'Mutasyon sonrası - Yukarı Ok'
WRITE(50,*)

700 END DO
END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! MUTATION ENDS !!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

IF (GEN.LT.GEN_NUM) THEN  ! No mutation after the last generation

DO i=1,pop_num
DO j=1,totstr_length
birey(i,j)=birey_ara(i,j)
END DO
END DO

END IF

200 END DO ! Main generation loop

CLOSE(20)
CLOSE(30)
CLOSE(40)
CLOSE(50)
CLOSE(70)
CLOSE(80)

17 FORMAT(i3,100f10.2)
22 FORMAT(i3,100f10.3)

! Calls "post_simulasyon" subroutine
CALL post(bestever_x,ch,run_say)
```

```
bit_time=time()
tot_time=bit_time-bas_time
tot_saat=tot_time/(60*60)
tot_dak=(tot_time-tot_saat*60*60)/(60)
tot_san=(tot_time-tot_saat*60*60-tot_dak*60)
WRITE (10,*) 'Toplam run zamanı=',tot_saat,'saat',tot_dak,'dakika',tot_san,'saniye'
CLOSE(10)

100 ENDDO !run_say

OPEN (90,FILE='sonuclar\\sonuclar'//ch//'\best_all.txt')
OPEN (100,FILE='sonuclar\\sonuclar'//ch//'\avg_all.txt')
OPEN (110,FILE='sonuclar\\sonuclar'//ch//'\var_all.txt')

DO I=1,gen_num
WRITE (90,17) I,(best_all(I,J),J=1,total_run)
WRITE (100,17) I,(avg_all(I,J),J=1,total_run)
WRITE (110,22) I,(var_all(I,J),J=1,total_run)
END DO

END program GENETIK_INJLOCX
```

Fonksiyon.f90

```
FUNCTION func(x,p,x_num,ch,inj_coef,run_say,penalty_coef) RESULT (cost)
USE DFLIB
USE DFPORT
IMPLICIT NONE
REAL(8) :: cost, f1, f2, g,Q(20)
INTEGER :: p,x_num,penalty_coef,A(20),LXY(20,20),I,Z,SAYAC,run_say
INTEGER(2) result
REAL(8) ::cp(10),heads(60,60)
REAL  x(1000,1000),HFB_K(1000),pen,total_q,totalnet_q,inj_coef
INTEGER tx,HFB(1000),HFB_LXY(1000,1000)
CHARACTER*5 :: ch

! This part reads *.wel file.
OPEN (UNIT=7,FILE="101_inj03.wel")
READ (7,*) (A(I),I=1,2)
READ (7,*) (A(3))
DO I=1,A(1)
READ(7,*) (LXY(I,Z),Z=1,3),Q(I)
END DO
CLOSE (UNIT=7,STATUS='KEEP')

! According to runno optimizes well rates or location in an iterative manner.
IF (MOD(run_say,3).EQ.1) THEN

DO I=1,A(1)
Q(I)=-1*x(p,I)
END DO
Q(6)=x(p,6)
Q(7)=x(p,7)
Q(8)=x(p,8)

! According to runno, optimizes well rates or location in an iterative manner.
ELSE IF (MOD(run_say,3).EQ.2) THEN

DO I=1,3
LXY(I+5,2)=NINT(x(p,2*I-1)) !injection well y-axes
LXY(I+5,3)=NINT(x(p,2*I)) !injection well x-axes
END DO

ELSE IF (MOD(run_say,3).EQ.0) THEN

DO I=1,A(1)
Q(I)=-1*x(p,I)
END DO

Q(6)=x(p,6)
Q(7)=x(p,7)
Q(8)=x(p,8)

END IF

! Modifies *.wel file.
OPEN (UNIT=7,FILE="101_inj03.wel")
WRITE (7,*),A(1),A(2)
WRITE (7,*),A(3)
DO I=1,A(1)
WRITE (7,21),(LXY(I,Z),Z=1,3),Q(I)
21 FORMAT (3i,f12.2)
END DO
CLOSE (UNIT=7,STATUS='KEEP')
```

```
! Calls MODFLOW.
RESULT = RUNQQ('c:\modflow\\mf2k.exe','101_inj03.nam')

! Modifies *.hed file.
OPEN (UNIT=8,FILE="101_inj03.hed")
DO I=1,30
READ(8,*) (HEADS(I,Z),Z=1,60)
END DO
CLOSE (UNIT=8,STATUS='KEEP')

! Calculate penalty values using
! the potential values at control points.
cp(1)=(8.0078-heads(15,23))
cp(2)=(8.0078-heads(9,29))
cp(3)=(8.0078-heads(21,34))
cp(4)=(8.0078-heads(6,40))
cp(5)=(8.0078-heads(15,42))

g=0
IF (MOD(run_say,3).EQ.1) THEN
DO I=1,5
g = g+MAX(0.0,cp(I))*MAX(0.0,cp(I))
END DO

ELSE IF (MOD(run_say,3).EQ.2) THEN
DO I=1,5
g = g+MAX(0.0,-cp(I))*MAX(0.0,-cp(I))
END DO

ELSE IF (MOD(run_say,3).EQ.0) THEN
DO I=1,5
g = g+MAX(0.0,cp(I))*MAX(0.0,cp(I))
END DO
END IF

OPEN(70,file='sonuclar\\sonuclar'//ch//'\penaltilar.txt')
WRITE(70,*)  SAYAC,g*penalty_coef
SAYAC=SAYAC+1

! Objective function
f1=0
DO I=1,A(1)-3
f1=f1-Q(I)
f2=0
END DO

DO I=A(1)-2,A(1)
f1=f1-inj_coef*Q(I)
END DO

IF (MOD(run_say,3).EQ.1) THEN
cost = MAX(0.0,f1 - penalty_coef * g)
ELSE IF (MOD(run_say,3).EQ.2) THEN
cost = MAX(0.0,g)
ELSE IF (MOD(run_say,3).EQ.0) THEN
cost = MAX(0.0,f1 - penalty_coef * g)
END IF

total_q=0
DO I=1,5
total_q=total_q+Q(I)
```

```
END DO

totalnet_q=0
DO I=1,x_num
totalnet_q=totalnet_q+Q(I)
END DO

WRITE (10,8) p,'.',(x(p,i),i=1,x_num),-total_q,-totalnet_q,cost,g
8 FORMAT (i,a,30f12.2)

RETURN
END FUNCTION
```

Post_Simulasyon.f90

```
SUBROUTINE  Post(bestever_x,ch,run_say)
USE DFLIB
USE DFPORT
IMPLICIT NONE
INTEGER :: A(20),LXY(20,20),I,Z,d,run_say
INTEGER(2) result
REAL bestever_x(30),heads(100,100),KSILON(100,100),HF(100,100)
REAL ZH(100,100),faytoe,sigma,pfre,pden
REAL DELY,DELX,Q(20)
INTEGER host
CHARACTER*5 :: ch

DELX=7000./60
DELY=100.

! This part reads *.wel file.
OPEN (UNIT=7,FILE="101_inj03.wel")
READ (7,*) (A(I),I=1,2)
READ (7,*) (A(3))
DO I=1,A(1)
READ(7,*) (LXY(I,Z),Z=1,3),Q(I)
END DO
CLOSE (UNIT=7,STATUS='KEEP')

!  Decision variables are assigned to Q values.
IF (MOD(run_say,3).EQ.1) THEN
DO I=1,A(1)
Q(I)=-1*bestever_x(I)
END DO

Q(6)=bestever_x(6)
Q(7)=bestever_x(7)
Q(8)=bestever_x(8)

ELSE IF (MOD(run_say,3).EQ.2) THEN
do I=6,8
LXY(I,2)=NINT(bestever_x(2*(I-5)-1)) ! Injection well x-axes
LXY(I,3)=NINT(bestever_x(2*(I-5))) ! Injection well y-axes
end do

ELSE IF (MOD(run_say,3).EQ.0) THEN

DO I=1,A(1)
Q(I)=-1*bestever_x(I)
END DO
Q(6)=bestever_x(6)
Q(7)=bestever_x(7)
```

```
Q(8)=bestever_x(8)

END IF

! This part modifies *.wel file
OPEN (UNIT=7,FILE="101_inj03.wel")
WRITE (7,*),A(1),A(2)
WRITE (7,*),A(3)
DO I=1,A(1)
WRITE (7,21),(LXY(I,Z),Z=1,3),Q(I)
21  FORMAT (3i,f12.2)
END DO
CLOSE (UNIT=7,STATUS='KEEP')

! This part calls MODFLOW
RESULT = RUNQQ('c:\modflow\\mf2k.exe','101_inj03.nam')

! Calculate the fay toe value
pfre=1.
pden=1.025
sigma=(pden-pfre)/pfre
d =25
faytoe=(1+sigma)*sigma*d*d/2


  ! Modifies *.hed file
OPEN (UNIT=8,FILE="101_inj03.hed")
DO I=1,30
READ(8,*) (HEADS(I,Z),Z=1,60)
END DO

DO I=1,30
DO Z=1,60
IF (HEADS(I,Z).LE.0) THEN
HEADS(I,Z)=0.
END IF
END DO
END DO

DO I=1,30
DO Z=1,60
IF (HEADS(I,Z).GT.FAYTOE) THEN
KSILON(I,Z)=d
ZH(I,Z)=0
HF(I,Z)=SQRT(2*HEADS(I,Z)+(1+SIGMA)*D*D)

ELSE
KSILON(I,Z)=SQRT(2*HEADS(I,Z)/SIGMA/(1+SIGMA))
ZH(I,Z)=d-ksilon(I,Z)
HF(I,Z)=SQRT(2*sigma*HEADS(I,Z)/(1+sigma))+d
END IF

END DO
END DO
CLOSE (UNIT=8,STATUS='KEEP')

! Calculates freshwater head and interface depth using the potential values.
OPEN(23,file='sonuclar\\sonuclar'//ch//'\wells.wel')

WRITE (23,*),A(1),A(2)
WRITE (23,*),A(3)
DO I=1,A(1)
```

```
WRITE (23,21),(LXY(I,Z),Z=1,3),Q(I)
END DO
CLOSE (UNIT=23,STATUS='KEEP')

OPEN(17,FILE='sonuclar\\sonuclar'//ch//'\ZH.hed')
WRITE (17,13) 0.00,(DELX*(Z-1)+DELX/2,z=1,60)
DO I=1,30
WRITE (17,13) DELY*(I-1)+DELY/2,(ZH(I,Z),Z=1,60)
13  format (1000f8.2)
END DO
CLOSE (UNIT=17,STATUS='KEEP')

OPEN(19,FILE='sonuclar\\sonuclar'//ch//'\HF.hed')
WRITE (19,13) 0.00,(DELX*(Z-1)+DELX/2,z=1,60)
DO I=1,30
WRITE (19,13) DELY*(I-1)+DELY/2,(HF(I,Z),Z=1,60)
END DO
CLOSE (UNIT=19,STATUS='KEEP')


OPEN(21,FILE='sonuclar\\sonuclar'//ch//'\FAY.hed')
WRITE (21,13) 0.00,(DELX*(Z-1)+DELX/2,z=1,60)
DO I=1,30
WRITE (21,13) DELY*(I-1)+DELY/2,(HEADS(I,Z),Z=1,60)
END DO
CLOSE (UNIT=19,STATUS='KEEP')
```

END subroutine post

APPENDIX B

SA CODE

Main_SA.f90

```
PROGRAM KORKUT_SA
INTEGER SEED,I,J,K,x_num,run_say,total_run,k1,runno,t_update,c
REAL x(30),xnew(30),best_x(30),f,fnew,fmax,up_bound(30)
REAL low_bound(30),Step_length(30),T,T_coef,T0 !Maximum dec.variable is 30.
REAL penalty_coef,ratio,inj_coef
INTEGER num_accepted,num_accepteDim(30),num_unaccepted,
INTEGER num_ofbounds,num_better,axiter_say,Outiter_say,Initer_say,totiter_say
INTEGER Variable_no
CHARACTER*3 :: cg
CHARACTER*5 :: ch

totiter_say=0
seed=12312586
bas_time=time()

OPEN(UNIT=17,FILE='c:\Korkut_GA\\RUNNO.txt')
READ (17,*) RUNNO
CLOSE (17)
OPEN(UNIT=18,FILE='c:\Korkut_GA\\RUNNO.txt',status='replace')
WRITE (18,*) RUNNO+1
CLOSE(18)
WRITE (ch,2000) RUNNO
2000  FORMAT(I5)

!  Reads input file
OPEN(5,file='input.txt')
READ (5,*) total_run
READ (5,*) x_num
DO k1=1,x_num
READ (5,*) low_bound(k1)
READ (5,*) up_bound(k1)
ENDDO
DO k1=1,x_num
READ (5,*) x(k1)
ENDDO
DO k1=1,x_num
READ (5,*) Step_length(k1)
ENDDO
READ (5,*) T
READ (5,*) T_coef
READ (5,*) Initer_say
READ (5,*) Outiter_say
READ (5,*) maxiter_say
READ (5,*) c
READ (5,*) penalty_coef
READ (5,*) inj_coef

RESULT = MAKEDIRQQ('sonuclar\sonuclar'//ch//'')

OPEN(10,file='sonuclar\\sonuclar'//ch//'\output.txt')
```

```
WRITE(10,*) 'PROGRAM CIKTILARI'
OPEN(20,file='sonuclar\\sonuclar'//ch//'\f.txt')
WRITE(20,*) 'f degerleri'
OPEN(30,file='sonuclar\\sonuclar'//ch//'\eniyi_f.txt')
WRITE(30,*) 'EN İYİ DEGERLER'
WRITE(30,*) 'Iterasyon sayısı','En iyi cost','Sicaklik'
OPEN(40,file='sonuclar\\sonuclar'//ch//'\kontrol.txt')
OPEN(50,file='sonuclar\\sonuclar'//ch//'\steps.txt')
OPEN(60,file='sonuclar\\sonuclar'//ch//'\eniyi_xset.txt')
WRITE (10,*) 'Total run sayisi=',total_run
WRITE (10,*) 'Değişken sayisi=',x_num
DO k1=1,x_num
WRITE (10,*) 'Low bound=',low_bound(k1)
WRITE (10,*) 'Upper bound=',up_bound(k1)
ENDDO
DO k1=1,x_num
WRITE (10,*) 'Variable',k1,x(k1)
ENDDO
DO k1=1,x_num
WRITE (10,*) 'Initial Steplengths=',Step_length(k1)
ENDDO

WRITE (10,*) 'Initial Temperature=',T
WRITE (10,*) 'Temperature coefficient=',T_coef
WRITE (10,*) 'Inner iteration number=',Initer_say
WRITE (10,*) 'Outer iteration number=',Outiter_say
WRITE (10,*) 'Maximum iteration number=',maxiter_say
WRITE (10,*) 'Steplength adjustment coeff.=',c
WRITE (10,*) 'Penalty coefficient=',penalty_coef
WRITE (10,*) 'Seed number=',seed
WRITE (10,*) 'Inj coef=',inj_coef
WRITE (10,*)
WRITE (10,*)
WRITE (10,*)

!!!!!!!!!!!!!!!!!!MAIN LOOP!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
DO 100 run_say=1,total_run
totiter_say=0
seed=seed-100
num_accepted=0
num_unaccepted=0
num_ofbounds=0
num_better=0
DO I=1,x_num
num_accepteDim(I)=0
ENDDO
T0=T

CALL SRAND(seed)
totiter_say=totiter_say+1
f = func(x,x_num,ch,penalty_coef,totiter_say,inj_coef)
DO i=1,x_num
WRITE(10,*) 'xint=',x(i),'fint=',f
ENDDO
WRITE(10,*)
WRITE(20,*) totiter_say,f

fmax=f
DO k1=1,x_num
best_x(k1)=x(k1)
ENDDO
WRITE(30,*) totiter_say,fmax,T
```

```
WRITE(50,*) 'xnew(i)-x(i)','   step_length(i)'

DO 325 K=1,Outiter_say
DO 225 J=1,Initer_say
DO 125 Variable_no=1,x_num

DO I = 1, x_num
IF (I.EQ.Variable_no) THEN
! Step length is multiplied with a random number btwn. -1 and 0.
xnew(i)=x(i)+(RAND()*2-1)*step_length(i)
ELSE
xnew(i)=x(i)
ENDIF

! If new point is out of boundaries, get a new one.
IF((xnew(i).LT.low_bound(i)) .OR. (xnew(i).GT.up_bound(i))) THEN
xnew(i)=low_bound(i)+(up_bound(i)-low_bound(i))*RAND()
num_ofbounds=num_ofbounds+1
ELSE
ENDIF
WRITE(50,*) xnew(i)-x(i),step_length(i)
END DO
WRITE(50,*)
totiter_say=totiter_say+1

! Call function subroutine.
fnew = func(xnew,x_num,ch,penalty_coef,totiter_say,inj_coef)

! Control if the new solution is the best solution.
IF (fnew.GT.fmax) THEN ! New optimum
fmax=fnew
num_better=num_better+1
DO i=1,x_num
best_x(i)=xnew(i)
ENDDO
WRITE(30,*) totiter_say,fmax,T
ENDIF
DO i=1,x_num
WRITE(10,*) 'xnew=',xnew(i),'fnew=',fnew
ENDDO
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! If the new solution is a better solution, accept it.
IF (fnew.GT.f) THEN
num_accepted=num_accepted+1
num_accepteDim(Variable_no)=num_accepteDim(Variable_no)+1
WRITE(10,*) 'Since fnew>f (Bttr thn prvs sltn, f=fnew, The new solution is ACCEPTED'
f=fnew
DO k1=1,x_num
x(k1)=xnew(k1)
ENDDO
ELSE     ! Else, accept according to Metropolis criteria.
r=RAND()
IF (r.le.EXP(-(f-fnew)/T)) THEN
WRITE(10,*) 'fnew<f, but since random num.',r,' is less than exp-(f-fnew)/T)=',    &
EXP(-(f-fnew)/T),'f=fnew, The new solution is accepted'
WRITE(10,*) 'Temperature T=',T,'f-fnew=',f-fnew
num_accepted=num_accepted+1
num_accepteDim(Variable_no)=num_accepteDim(Variable_no)+1
f=fnew
DO k1=1,x_num
x(k1)=xnew(k1)
```

```
ENDDO
ELSE
WRITE(10,*) 'Since fnew<f and r<exp(...), f=f  The new soltn is not accepted'
num_unaccepted=num_unaccepted+1
ENDIF
ENDIF
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

DO i=1,x_num
WRITE(10,*) 'x=',x(i),'f=',f
ENDDO
WRITE(10,*)
WRITE(20,*) totiter_say,f

125 END DO
225 END DO

WRITE(40,*) 'Temperature=',T
WRITE(40,*) 'Temperature update sayisi=',t_update
WRITE(40,*) 'The number of better solutions=',num_better
WRITE(40,*) 'The number of accepted but worse solutions=',num_accepted
WRITE(40,*) 'The number of unaccepted and worse solutions=',num_unaccepted
WRITE(40,*) 'The best solution=',fmax

DO i=1,x_num
WRITE(40,*) 'best_x=',best_x(i)
ENDDO
WRITE(40,*)

! The ratio of the no. of acptd.  newcomers to the no of  total eval.  is to be around 0.5.
DO I=1,x_num
ratio=num_accepteDim(I)*1./Initer_say
WRITE(50,*) 'ratio for',i,num_accepteDim(I),Initer_say,ratio
IF (ratio.ge.0.6) then
step_length(I)=step_length(I)*(1+c*(ratio-0.6)/0.4)
ELSE IF (ratio.le.0.4) then
step_length(I)=step_length(I)/(1+c*(0.4-ratio)/0.4)
END IF
IF (step_length(I).gt.up_bound(I)+low_bound(I)) then
step_length(I)=up_bound(I)-low_bound(I)
END IF
END DO

! Update temperature.
T=T*T_coef
t_update=t_update+1

! Reset values
seed=seed-10000
num_accepted=0
num_unaccepted=0
num_ofbounds=0
num_better=0
DO I=1,x_num
num_accepteDim(I)=0
ENDDO

! After temperature update, continue with the best-ever solution.
f=fmax
DO i=1,x_num
x(i)=best_x(i)
ENDDO
```

```
325 END DO

WRITE(10,*) 'Temperature update sayısı=',t_update
WRITE(10,*) 'The number of better solutions=',num_better
WRITE(10,*) 'The number of accepted but worse solutions=',num_accepted
WRITE(10,*) 'The number of unaccepted and worse solutions=',num_unaccepted
WRITE(10,*) 'The best solution=',fmax,'found at iteration no'

DO i=1,x_num
WRITE(10,*) 'best_x=',best_x(i)
WRITE(60,*) 'best_x=',best_x(i)
ENDDO

100 ENDDO
!!!!!!!!!!!!!!MAIN LOOP!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
END PROGRAM
```

Fonksiyon.F90

```
FUNCTION func(x,x_num,ch,penalty_coef,totiter_say,inj_coef) RESULT (cost)
USE DFLIB
USE DFPORT
IMPLICIT NONE
REAL(8) :: cost, f1, f2, g,Q(20)
INTEGER :: p,x_num,penalty_coef,A(20),LXY(20,20),I,Z,SAYAC,run_say,totiter_say
INTEGER(2) result
REAL(8) ::cp(10),heads(60,60)
REAL  x(30),HFB_K(1000),pen,total_q,inj_coef
INTEGER tx,HFB(1000),HFB_LXY(1000,1000)
CHARACTER*5 :: ch

! This part reads the *.wel file
OPEN (UNIT=7,FILE="101_INJ03.WEL")
READ (7,*) (A(I),I=1,2)
READ (7,*) (A(3))
DO I=1,A(1)
READ(7,*) (LXY(I,Z),Z=1,3),Q(I)
END DO
CLOSE (UNIT=7,STATUS='KEEP')

! Decision variables are assigned to well values.
DO I=1,A(1)-3
Q(I)=-1*x(I)
END DO

Q(6)=x(6)
Q(7)=x(7)
Q(8)=x(8)

DO I=6,8
LXY(I,2)=NINT(x(2*I-3)) ! Injection well y-axes.
LXY(I,3)=NINT(x(2*I-2)) ! Injection well x-axes.
END DO

OPEN (UNIT=7,FILE="101_INJ03.WEL")
WRITE (7,*),A(1),A(2)
WRITE (7,*),A(3)
DO I=1,A(1)
WRITE (7,21),(LXY(I,Z),Z=1,3),Q(I)
21 FORMAT (3i,f12.2)
END DO
```

```
CLOSE (UNIT=7,STATUS='KEEP')

! Calls Modflow.exe.
result = RUNQQ('c:\modflow\\mf2k.exe','101_INJ03.nam')

! This part reads *.hed file.
OPEN (UNIT=8,FILE="101_INJ03.hed")
DO I=1,30
READ (8,*) (HEADS(I,Z),Z=1,60)
END DO
CLOSE (UNIT=8,STATUS='KEEP')

! Calculate penalty values using the potential values at control points.
cp(1)=(8.0078-heads(15,23))
cp(2)=(8.0078-heads(9,29))
cp(3)=(8.0078-heads(21,34))
cp(4)=(8.0078-heads(6,40))
cp(5)=(8.0078-heads(15,42))

g=0
DO I=1,5
g = g+max(0.0,cp(I))*max(0.0,cp(I))
END DO

OPEN (70,file='sonuclar\\sonuclar'//ch//'\penaltilar.txt')
WRITE(70,*)  totiter_say,g,g*penalty_coef

!  Objective function
f1=0
DO I=1,A(1)-3
f1=f1-Q(I)
f2=0
END DO

DO I=A(1)-2,A(1)
f1=f1-inj_coef*Q(I)
END DO

cost = max(0.0,f1 - penalty_coef * g)

RETURN
END FUNCTION
```

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Demirbaş, Korkut
Nationality: Turkish (TC)
Date and Place of Birth: 26 February 1978, Trabzon
Marital Status: Single
Phone: +90 312 221 11 89
email: korkutdemirbas@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Civil Engineering	2003
BS	METU Civil Engineering	1999
High School	Trabzon Anadolu High School	1995

WORK EXPERIENCE

Year	Place	Enrollment
2006- Present	TÜBİTAK	Assistant Expert
2003-2004	METU Department of Civil Engineering	Computer Assistant
1998 July	Karayolları Genel Müdürlüğü	Intern Engineering Student

FOREIGN LANGUAGES

Advanced English

PUBLICATIONS

1. Demirbaş, K., *Combined Optimization-simulation of an Excavation Site for Dewatering Purposes*, MS Thesis, Middle East Technical University, Civil Eng. Dept., Ankara, 2003.

2. Demirbas, K., Altan Sakarya, A. B., Onder, H., *Optimal Groundwater Management Using Evalutionary Algorithm To Prevent Saltwater Intrusion*, IAHR International Groundwater Symposium, 18-20 June, Istanbul, 2008.

3. Demirbas, K., Altan Sakarya, A. B., Onder, H., *Combined Simulation Optimization of an Excavation Site for Dewatering Purpose*, 8th International Congress on Advences in Civil Engineering, 15-17 September, Gazimagosa, 2008.

4. Demirbas, K., Altan Sakarya, A. B., Onder, H., *Combined Simulation Optimization of a Coastal Aquifer by Using Genetic Algorithm*, 6th International Symposium on Environmental Hydraulics, 23-25 June, Greece, 2010.

HOBBIES

3D Animation, Drawing, Nature and Biology, Movies, Scuba