WRC Research Report No. 218

GENETIC-ALGORITHM-BASED DESIGN OF GROUNDWATER QUALITY MONITORING SYSTEM

by

J. Wayland Eheart
Scott E. Cieniawski
S. Ranjithan

Department of Civil Engineering
University of Illinois at Urbana-Champaign

Project No. G-2017-02
ISSN 0073-5442

Water Resources Center
University of Illinois at Urbana-Champaign
205 North Mathews Avenue
Urbana, Illinois 61801

September 1993

The work on which this report is based was supported in part by funds provided by the United States Department of the Interior as authorized under the Water Resources Research Act of 1984. The contents of this report do not necessarily reflect the views and policies of the U.S. Department of the Interior, nor does mention of trade names or commercial products constitute their endorsement by the United States Government.

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Environmental Engineering in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign
1993

Scott Edward Cieniawski
B.S., University of Illinois at Urbana-Champaign, 1991

ABSTRACT

This research builds on the work of *Meyer and Brill* [1988] and subsequent work by *Meyer et al.* [1990], *Meyer et al.* [1992], and *Meyer* [1992] on the optimal location of a network of groundwater monitoring wells under conditions of uncertainty. A method of optimization is developed using genetic algorithms (GAs) which allows consideration of the two objectives of *Meyer et al*. [1992], maximizing reliability and minimizing contaminated area, separately yet simultaneously. The GA-based solution method can generate both convex and non-convex points of the tradeoff curve, can accommodate non-linearities in the two objective functions, and is not restricted to the peculiarities of a weighted objective function. Furthermore, GAs can generate large portions of the tradeoff curve in a single iteration and may be more efficient than methods that generate only a single point at a time.

Four multi-objective GAs formulations are investigated and their performance in generating the multi-objective tradeoff curve is evaluated for the groundwater monitoring problem using two example data sets. The GA formulations are compared to each other and to simulated annealing on both performance and computational intensity.

The simulated annealing based technique used by *Meyer et al.* [1992] relies on a weighted objective function which finds only a single point along the tradeoff curve for each iteration, while the multiple-objective GA formulations are able to find many convex and non-convex points along the tradeoff curve in a single iteration. Each iteration of simulated annealing is approximately five times faster than an iteration of the genetic algorithm, but several simulated annealing iterations are required to generate the tradeoff curve. GAs are able to find a larger number of non-dominated points on the tradeoff curve in a single iteration, and are therefore just as computationally efficient as simulated annealing in terms of generating the tradeoff curves.

None of the GA formulations demonstrate the ability to generate the entire tradeoff curve in a single iteration, but they yield either a good estimation of all regions of the tradeoff curve except the very highest and very lowest reliability ends or a good estimation of the high reliability end alone.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor J. Wayland Eheart, for all of his valuable advice. His insight and guidance provided this research effort with the required direction and focus. I would also like to thank my colleagues, S. Ranji Ranjithan, Brian Ritzel, and Pascal Storck, who provided suggestions from their related research. Finally, my thanks to the Water Resources Center at the University of Illinois for providing the funding for this research.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Chapter 1
Introduction

The solution to many environmental problems is not as simple as finding the single best solution to a given situation. There are likely to be many competing objectives that interact in complicated ways. Managers are forced to examine problems that incorporate multiple and often conflicting objectives. It becomes important, therefore, to provide these decision makers with tools and information that will allow them to make informed decisions on how much of a given objective must be sacrificed to obtain improvement in other objectives. An optimization tool that allows us to generate an entire tradeoff curve in a single iteration will be more useful to the decision-making process than methods, such as those discussed in *Cohon* [1978], that generate just a single point at a time.

This paper builds on the work of *Meyer and Brill* [1988] and subsequent work by *Meyer et al.* [1990] on the optimal location of a network of groundwater monitoring wells under conditions of uncertainty. Given an unknown hydraulic flow field surrounding a potentially leaky landfill their objective is to maximize the predicted reliability of the monitoring network, where reliability is determined by the percentage of Monte Carlo plume realizations that are detected by the monitoring network. *Meyer et al.* [1992] extend these works by considering an objective function that consists of a weighted sum of reliability and a contaminated area objective.

This research investigates a method of optimization using genetic algorithms (GAs) and considers the two objectives, maximizing reliability and minimizing contaminated area at the time of first detection, separately yet simultaneously. The GA-based solution method has a distinct advantage over traditional multi-objective programming approaches discussed by *Cohon* [1978], because it can generate both convex and non-convex points of the tradeoff curve, accommodate non-linearities in the two objective functions, and is not restricted by the peculiarities of a weighted objective function. *Steuer's* [1986] Tchebycheff method is also able to generate non-convex points on the tradeoff curve, but it relies on a weighting factor. Since the optimal point may remain unchanged over a wide range of weighting factors, weighted objective functions require proper scaling of the objective functions or the weighting factor in order to obtain good results. Furthermore, GAs have the ability to generate large portions of the tradeoff curve in a single iteration, while weighted objective functions generate only a single point per iteration.

In Chapter 2, we begin by introducing the groundwater monitoring problem. Chapter 3 contains an introduction to genetic algorithms, how they work, and a discussion of their strengths and weaknesses. Chapter 4 describes the more advanced GA operators used in solving the multi-objective problem. In Chapter 5, five GA codings are presented for generating the tradeoff curve for the groundwater monitoring problem and each is used to generate the tradeoff curve between reliability and contaminated area for two example data sets. Chapter 6 presents the results, and the performance of the formulations are compared to each other and to the results *Meyer et al.* [1992] obtained using simulated annealing. Finally, Chapter 7 presents some conclusions and identifies possible directions for future research.

Chapter 2
The Groundwater Monitoring Problem

The basic groundwater monitoring problem can be described as follows. Given a potential contaminant source and an unknown hydraulic conductivity field, where should monitoring wells be located to maximize the probability of leak detection while minimizing the area contaminated by the plume? The groundwork for this discussion has been already laid. *Massman and Freeze* [1987] use Monte Carlo simulation to take into account the uncertainty in the hydraulic conductivity field of an aquifer and determine the reliability of any given monitoring network. *Meyer and Brill* [1988] and *Meyer et al.* [1990] extend this work by seeking to optimally place the groundwater monitoring wells, under these conditions of uncertainty, in order to maximize the reliability of the monitoring network. The single-objective groundwater monitoring optimization problem they develop is an extension of *Church and Revelle's* [1974] maximal covering location problem (MCLP). *Meyer et al.* [1992] further extend these works by incorporating both a reliability and a contaminated area objective into the optimization problem. They refer to the two-objective problem as the extended p-median problem (EPMP) which they adapted from *Revelle and Swain's* [1970] integer formulation of the p-median problem.

The problem is depicted in Figure 1, adapted from *Meyer and Brill* [1988]. Monte Carlo simulation is used to generate several hundred plume realizations by randomly sampling from the assumed distributions of aquifer parameters and leakage events. During this simulation a data file is generated that contains the plumes that are detected by each potential monitoring well, and the amount of area that is contaminated when each well first detects each plume. Reliability is characterized by the percentage of simulated plumes that are detected by a network of wells. Since the number of simulated plumes is held constant this is the same as minimizing the number of undetected plumes. Contaminated area is measured by the area contaminated when a plume is first detected, summed over all plumes.

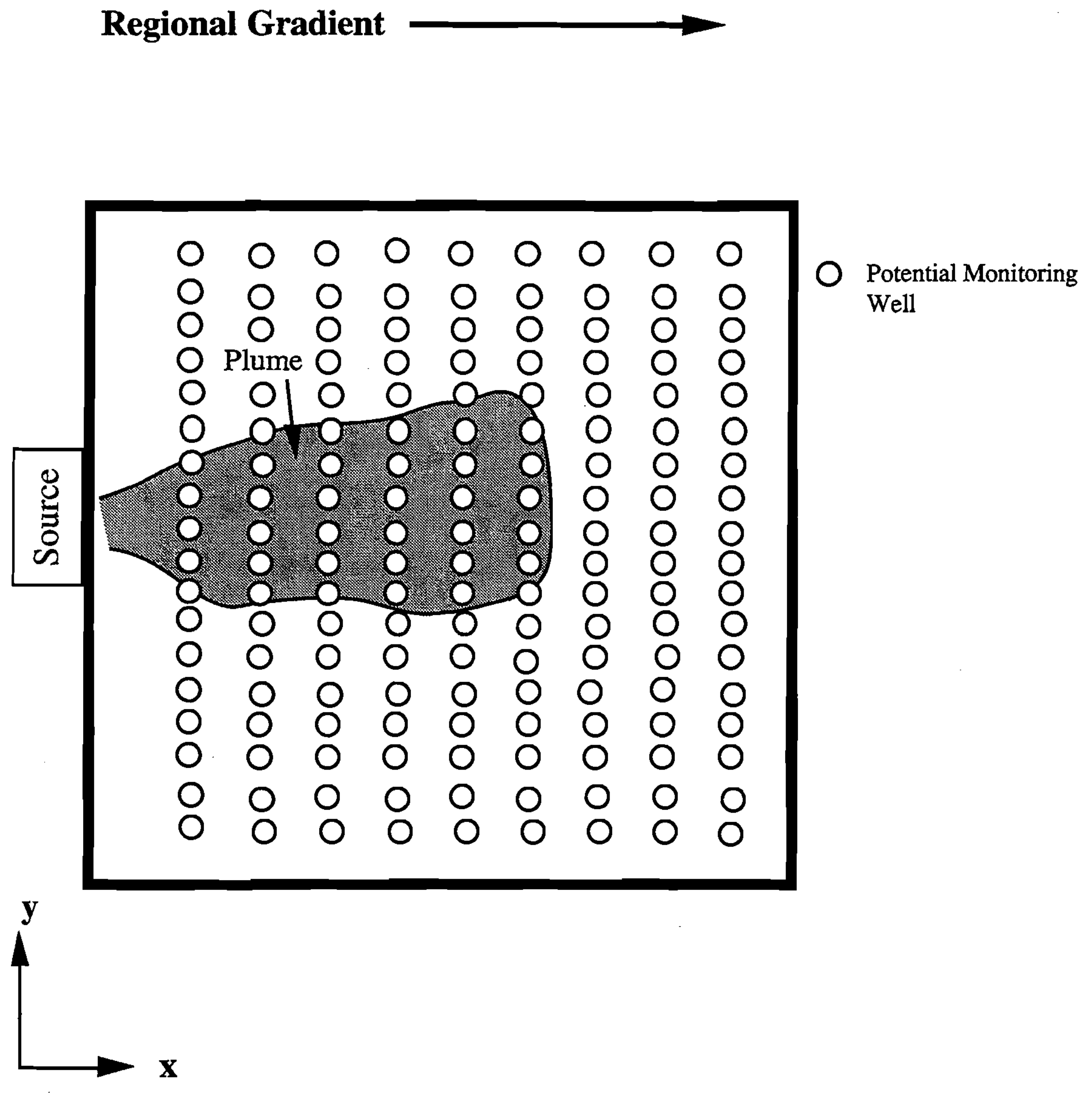

Figure 1. Monitoring Problem Layout (adapted from *Meyer and Brill*, 1988)

From this analysis we can develop the following multiple objective optimization problem that *Meyer et al.* [1992] adapted from *ReVelle and Swain's* [1970] integer formulation of the p-median problem for use on the groundwater monitoring problem:

$$\text{Minimize} \quad Z_1 = \sum_{i \in I} w_i \qquad (1)$$

$$\text{Minimize} \quad Z_2 = \sum_{i \in I} \sum_{j \in N_i} A_{ij} y_{ij} \qquad (2)$$

subject to:

$$w_i + \sum_{j \in N_i} y_{ij} \geq 1 \qquad \forall \; i \in I \qquad (3)$$

$$x_j \geq y_{ij} \qquad \forall \; i \in I, j \in N_i \qquad (4)$$

$$Qw_i + \sum_{j \in N_i} x_j \leq Q \qquad \forall \; i \in I \qquad (5)$$

$$\sum_{j \in J} x_j = Q \qquad (6)$$

$$x_j = (0,1) \qquad \forall \; j \in J \qquad (7)$$

$$y_{ij} = (0,1) \qquad \forall \; i \in I, \forall \; j \in J \qquad (8)$$

$$w_i = (0,1) \qquad \forall \; i \in I \qquad (9)$$

where:

I	=	set of distinct plumes.
J	=	set of potential well sites.
x_j	=	1 if a well is installed at site j; otherwise.
w_i	=	1 if plume i is undetected; otherwise.
y_{ij}	=	1 if plume i is initially detected at location j, otherwise.
A_{ij}	=	the area contaminated by plume i when it is first detected by potential well location j.
N_i	=	set of potential well locations, j, that detect plume i; if no wells detect plume i.
P	=	total number of simulated plumes
Q	=	the number of wells to be installed in the monitoring network.

The problem has two separate objectives, Z_1 and Z_2, both of which are to be minimized. Z_1 is the total number of undetected plumes. Minimizing the number of undetected plumes is equivalent to maximizing the reliability of the monitoring network. Z_2 is the total area contaminated by all detected plumes which *Meyer et al.* [1992] explain is equivalent to minimizing the average area of contamination by detected plumes.

Z_1 and Z_2 are competing objectives, which therefore necessitates the construction of a tradeoff curve of reliability vs. contaminated area. The tradeoff between reliability and contaminated area occurs because, in general, as the monitoring wells move further down gradient of the landfill the plume has more time to disperse. This dispersion increases the area of contamination, but allows detection by more potential well sites, hence, decreasing the number of undetected plumes for the same number of monitoring wells (i.e., Z_1 decreases as Z_2 increases). At potential well locations a large distance from the contaminant source both the reliability and the contaminated area objectives are degraded (*Meyer et al.* [1992]). The reason for this is that clean water in the aquifer dilutes the strength of the contaminant source and the concentration profiles of the contaminant a large distance from the leak fall below the hypothetical detection limit of the measurement apparatus.

Constraint (3) requires that each plume be detected by at least one well location or else go undetected. Constraint (4) prevents a plume from being detected at a well location that does not have an active well located at it. Constraint (5) requires that a plume be detected if there is an active well present in the network which can detect the plume. This constraint avoids the perverse situation of ignoring a plume if the resulting improvement in Z_2 would warrant the deterioration of Z_1. Constraint (6) limits the number of active well locations to Q, while constraints (7), (8), and (9) require that the decision variables be either zero or one. [*Meyer et al.*,1992].

Meyer et al. [1992] incorporate the two objectives from the EPMP formulation into a single weighted objective function corresponding to the expected cost of a contaminant leak, given by:

$$Z_3 = \sum_{i \in I} \sum_{j \in N_i} A_{ij} y_{ij} + \beta \sum_{i \in I} w_i \qquad (10)$$

where β is the cost of any undetected plume simulation. This optimization problem has a large number of Pareto-optimal solutions (i.e., no other solution performs better on both objectives), each corresponding to a unique range of β, that will form a tradeoff curve of reliability versus contaminated area. By changing the weighting factor for the cost of an undetected plume *Meyer et al.* [1992] are able to generate a tradeoff curve of reliability versus expected contaminated area for the EPMP.

Meyer and Brill [1988] were able to solve the MCLP using traditional linear programming techniques since the solutions, without the integer restrictions, were often all integer. However, this is not true of the EPMP, as *Meyer et al.* [1992] found the EPMP intractable to solve using the "conventional" branch-and-bound-linear-programming approach. They successfully apply simulated annealing to solve the EPMP. Simulated annealing, however, is limited to using a weighted sum objective function like the one presented above. A major shortcoming of their weighted-objective method is that it misses all non-convex portions of the tradeoff curve due to the linear combination of the two objectives.

Steuer [1986] presents the Tchebycheff weighting method that allows for the generation of convex and non-convex portions of a tradeoff curve by measuring the weighted distance in objective space from new solutions to some ideal point. The resulting Tchebycheff objective function is:

$$\text{Minimize:} Z_4 = \text{Max}\left\{(1-\beta)(Z_1^{**}-Z_1)\ ,\ \beta(Z_2^{**}-Z_2)\right\} \qquad (11)$$

where Z_1^{**} and Z_2^{**} are the values obtained by separately solving the EPMP optimization problem for optimal Z_1 and Z_2.

Both of these weighting methods suffer from shortcoming(s) common to most weighting methods; (1) the optimal point does not change over a wide range of weighting factor values, and/or (2) the optimal point changes several times over a small range of weighting factor values thus requiring the successive solution of the problem for a large number of different values of the weighting factor.

This paper investigates the ability of genetic algorithms to generate the entire tradeoff curve for the EPMP in a single iteration, following the hypothesis that GA's have the power to consider both objective functions separately yet simultaneously.

Chapter 3

Introduction to Simple Genetic Algorithms

3.1 Basic Operators of Genetic Algorithms

Genetic Algorithms (GAs) are a search technique developed by *Holland* [1975], that uses the mechanisms of natural selection to search through decision space for optimal solutions [*Goldberg*, p. 1]. GAs have been shown to be valuable tools for solving complex optimization problems in a broad spectrum of fields, including recent papers by *Ranjithan et al.* [1992], *Wang* [1991], and *McKinney and Lin* [1992] in the field of water resources.

GAs consist of three basic operations:

1. Selection
2. Crossover (Mating)
3. Mutation

In using genetic algorithms, several "strings" (usually binary vectors) are formed which represent different decision sets. These strings are evaluated on their performance (or "fitness") with respect to some objective function(s). Using this fitness value the strings compete in a **selection** tournament where strings having high fitness values are more likely to enter the mating population and strings with low fitness values are less likely to mate. The mating strings are randomly assigned a mating partner from within the mating population and a random crossover location is selected on the strings. Genetic information is exchanged between the two parent strings (**crossover**) to form children as shown in Figure 2. The parents are usually, but not always, deleted from the population and replaced in the population by the children to keep a stable population size. Mating between two strings takes place with a probability of P_{cross}. If mating does not take place the parent strings survive into the next generation.

Genetic algorithms are a very aggressive search technique and might quickly converge to a local optimum if the only components operating were selection and crossover. This is because GAs rapidly weed out stings with poor fitness values until all the strings of a population are identical, and in doing so may lose some important genetic information. Therefore, in order to

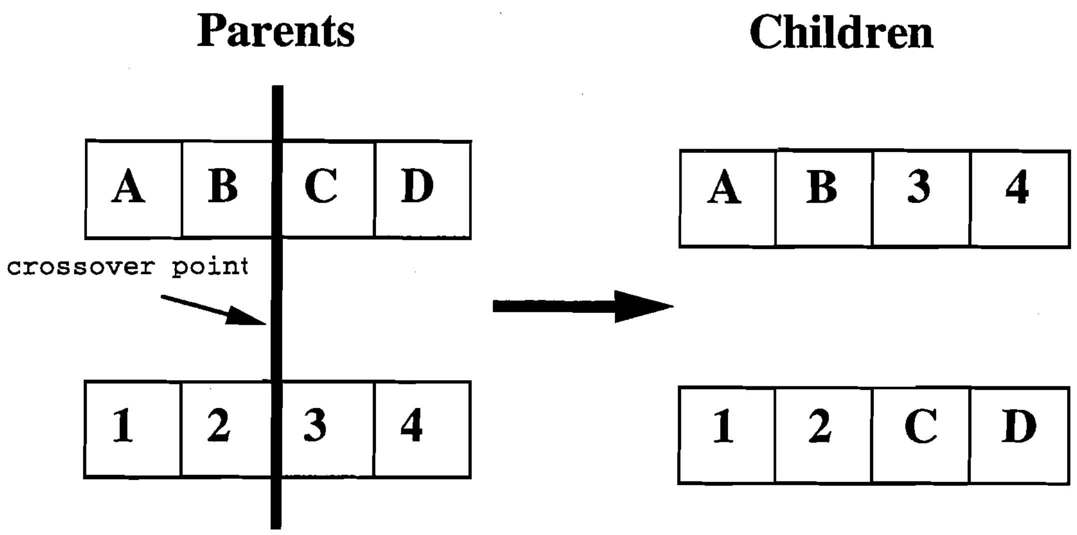

Figure 2. GA Crossover Operation

maintain some diversity in the string population some of a string's alleles are randomly **mutated**, with a probability of P_{mute}, to keep a population from converging too quickly. In Figure 2 each of the strings contains four alleles, the individual information locations on a string . The process of selection, crossover, and mutation is repeated for many generations in hopes of improving the performance of the population.

The basic outline of a simple GA computer formulation is given below:

```
Start Program
    Create Initial Population of Decision Strings
    Loop for N Generations
        Calculate String Fitness Function
        Select Top Performing Strings
        Shuffle Population to Create Random Mating Pairs
        Perform Crossover Operation
        Randomly Mutate String Alleles
    End Loop
    Output Results
End Program
```

The general theory behind this process is that strings with a high fitness values contain allele groupings ("building blocks") that are important to optimizing the objective function. By exchanging important building blocks between two strings that perform well, the GA attempts to produce children strings which contain the important building blocks from both parents and, therefore, perform even better than the parent strings. In this way GAs use Darwin's "survival of the fittest" theory to search through a decision space for an optimal solution. It is through this process of assembling strings with important building blocks that an optimal solution is found. This process also renders GAs intrinsically more computationally efficient than total enumeration.

3.2 Crossover and Selection Schemes

Goldberg [1989, p. 102-120] describes many different methods of performing the GA crossover operation. The most straightforward method is the single-point crossover illustrated in Figure 2. In single point crossover the same crossover location is selected on both parent strings. Two child strings are then created from the two parents strings, each containing information from both of the parent strings.

The selection scheme used in this research is binary tournament selection. Binary tournament selection is a very aggressive type of selection which *Goldberg and Deb* [1989] found helps eliminate the random noise from the selection process and improves the efficiency of the GA search algorithm. Under binary tournament selection each string in the current population is given two copies in the tournament population. The tournament population is then shuffled in order to create random tournament pairings. For each tournament pair the two strings compete directly with each other, and the string with the best fitness function survives into the mating population. The tournament loser is eliminated from the population. In this way the best performing of all strings will win both of its tournaments and be represented twice in the mating population. The worst performing of all strings will lose both of its tournaments and will be eliminated from the population. If two strings involved in a tournament have the same fitness value, the tie is broken randomly.

3.3 Genetic Algorithm Shortcomings

GAs generally have a shortcoming in regards to handling constraints. The only way that constraints may be included in a mathematical sense is by a formulation which uses the length of the string as a constraint, or which limits the values placed into each of the strings' allele locations. More commonly, penalty functions [*Goldberg*, 1989, p. 85-86] are used to assign a poor fitness value to strings that violate constraints. This implies an inherent weakness of the method in accommodating multiple constraints.

In the problem addressed here, Constraints (3) through (9) are intrinsically insured by the coding method and the objective function evaluation. Only feasible solutions are allowed in the initial population, the string length limits the number of active monitoring wells, and the objective function evaluations consider all detected plumes. There would be a problem in handling either of the objective functions as a constraint, but our intention, as stated above, is to consider the objective functions simultaneously in an attempt to determine the entire tradeoff curve at once.

Another general shortcoming of the GAs is the large number of objective function evaluations it must perform. For each generation the performance of each string must be evaluated. This function evaluation is very time-consuming and is often the limiting operation for the genetic algorithm. This bottleneck may be removed by performing the function evaluations in a parallel fashion since these evaluations are not interdependent. GAs are by nature, highly parallel, but have traditionally been run on serial machines only because of the limited availability of parallel computers.

A third shortcoming of genetic algorithms is that they are a heuristic search technique and are not theoretically guaranteed to find the optimal solution to a given problem. However, GAs are normally applied to problems, such as the EPMP, that are intractable using solution techniques that guarantee the theoretical optimum. GAs' performance are documented on a wide range of difficult problems in many different fields. [*Goldberg*, 1989, p. 125-142]

Chapter 4
Advanced GA Operators

Simple GAs with their three basic operators have proved to be a powerful tool for solving many single objective problems regardless of the problem's complexity. An area of GAs which has recently attracted attention is their ability to generate a tradeoff curve for a multi-objective optimization problem. The rest of this chapter describes the research that has been done in the area of multi-objective optimization and introduces the advanced GA operators that are needed in addition to selection, crossover, and mutation in order to generate multi-objective tradeoff curves.

4.1 Multiple Objective GAs

Two techniques have been suggested to allow GAs to incorporate multi-objective optimization problems; vector evaluated GAs and Pareto-optimal ranking GAs. Although previous research has found that these techniques produce some promising preliminary results, the methods have not yet been fully investigated. These two methods are described below along with a method that attempts to capitalize on the individual strengths of each of the two techniques by combining the two approaches into a single formulation.

4.1.1 Vector Evaluated GA Formulation

This formulation was introduced by *Schaffer* [1985] who refers to it as the Vector Evaluated Genetic Algorithm (VEGA). This formulation derives its fitness functions directly from the objective functions Z_1 and Z_2. During the tournament selection process, this formulation randomly selects one or the other objective, with a specified probability, to determine the binary tournament winner. P_{rel} is the probability that the winner of the tournament will be determined by the string with the highest reliability score. Correspondingly, $(1 - P_{rel})$ is the probability that the winner is selected based on lowest average contaminated area.

For each tournament pair the fitness function is chosen randomly. Therefore, the population at each generation contains strings which perform well in the reliability objective and those which perform well with regard to the contaminated area objective. It is hoped that through cross-mating between the two groups a population of strings will evolve that performs well on both objective functions. Care must be exercised not to pressure the population towards the knee of the tradeoff curve by only cross-mating. Therefore, a string's mating partner is randomly chosen, and it is possible that two strings that perform well on the same attribute will be partners.

4.1.2 Pareto-Optimal Ranking Formulation

The ranking scheme discussed by *Goldberg* [1989, p. 201] uses the idea of Pareto-optimality to assign a fitness function to each GA string. More specifically it uses the ideas of Pareto domination and inferiority to determine which strings outperform other strings. For a multiple objective problem with J objectives, and assuming maximization of all objectives without loss of generality, solution A dominates solution B if:

$$Z_{Aj} \geq Z_{Bj} \qquad \forall\, j \in J$$

and $$Z_{Aj} > Z_{Bj} \qquad \text{for at least one } j \in J$$

where,

Z_{ij} = the objective function value of solution i for objective j

If a given solution is not dominated by any other solutions it is said to be non-dominated. A solution is inferior if at least one solution dominates it.

To incorporate the Pareto-optimal ranking scheme, the current population is searched and all non-dominated solutions are given a rank of 1. These solutions are then temporarily removed from the population and the population is searched again. This time all non-dominated solutions are given a rank of 2. The procedure continues until all solutions have been ranked. Figure 3 graphically illustrates the rank of several solutions (shown in objective space) for the

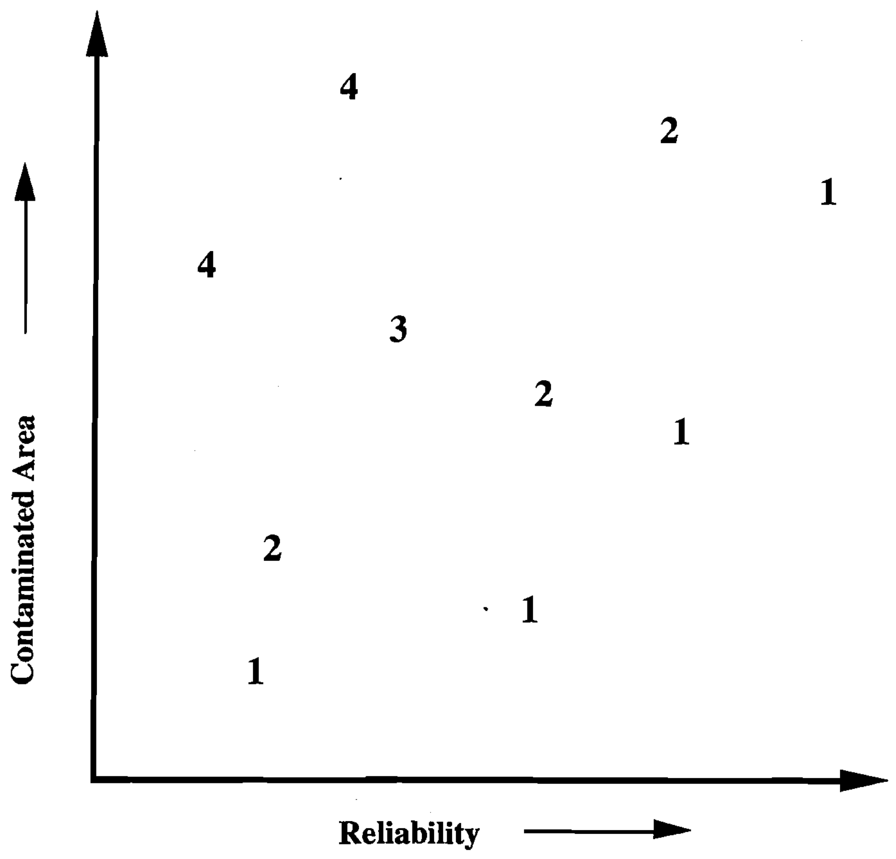

Figure 3. Graphical Representation of Pareto-Optimal Ranking Scheme

groundwater monitoring problem where reliability is to be maximized and average contaminated area is to be minimized. *Liepins et al.* [1988] found that this formulation outperforms VEGA in the solution of a constrained set covering problem.

4.1.3 Combination of the VEGA and Ranking Formulations

The third formulation is based on a combination of the VEGA and the Pareto-optimal ranking formulations. It operates as follows:

1. For the first N_{change} generations use a VEGA selection criteria with the probability of selecting for reliability, P_{rel}, set between 0.50 and 0.90.
2. After N_{change} generations switch to the Pareto-optimal ranking formulation.

This formulation is an attempt to make use of the strengths of both the VEGA and Pareto optimal ranking formulations. VEGA allows the programmer to force the search towards a certain portion of the tradeoff curve, in this case the high reliability end, achieved by a high value for P_{rel}. By switching back to Pareto-optimal ranking the GA is then able to search the Pareto-optimal space and trickle back down the tradeoff curve.

Care must be taken to insure that the population does not converge to a single optimum before switching selection schemes. In our problem we noticed that the GA population would begin to lose much of its genetic diversity by approximately the 70th generation, so we investigate N_{change} values around 70. This value must be selected individually for each GA application since the speed of convergence seems to be correlated to the length of the strings and population size [*Goldberg et al.*, 1992].

4.2 Maintaining Diversity in the GA Population using Sharing

As the GA's population of strings proceeds through a large number of generations it begins to lose its genetic diversity which is vital to the GA's problem solving ability. After a large number of generations the population will converge to a single solution. Mutation can help maintain some diversity, but mutation is random and cannot assure that a diverse set on good building blocks is maintained. An advanced GA operator called sharing was introduced by *Goldberg and Richardson* [1987] as a means of maintaining a stable population of good building blocks.

Sharing is based on the idea of niche formation in the natural world. Species which depend on the same resources for survival occupy the same niche. These species must then compete with each other for the resources represented by that particular niche. As more and more species begin to occupy a particular niche the chances of survival of each individual is degraded. This forces a diversity of species that each make use of different resources. By forming many different niches and limiting the number of individuals in a particular niche, the natural world is able to maintain a diverse range of species that make optimal use of all available resources.

The idea of GA sharing works in the same manner. Strings that are very similar to one another degrade each other's fitness function and, therefore, decrease each other's chance of survival. By degrading the fitness function of similar strings, sharing forces diversity in the GA population by rewarding the dissimilar strings. *Goldberg and Richardson* [1987] and *Deb and Goldberg* [1989] use sharing GAs to maintain a stable population of diverse strings that find all peaks of a multi-modal, single-objective test function.

4.3 Implementation of Sharing

To implement sharing in GAs a distance measure is used to determine the similarity of strings in the population (the distance measure is chosen individually for each problem). The parameter σ_{share} is used to set the maximum distance between strings at which sharing occurs. Strings that are a less than a distance of σ_{share} away from each other are said to occupy the same niche and degrade each other's fitness function. Strings that are more than a distance of σ_{share} away from each other occupy different niches and do not degrade each other's fitness function.

Sharing may be applied to the GA fitness function using a sharing function such as *Goldberg and Richardson's* [1987] triangular sharing function shown in Figure 4. Each string is compared to every other string in the current population, and a given string's fitness function is now penalized in proportion to how similar it is to the rest of the strings in the population. A string's new fitness is now given as:

$$f_{new}(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} share(d(x_i,x_j))} \qquad \text{(from \textit{Goldberg} [1989, p. 192])} \qquad (12)$$

where $f_{new}(x_i)$ is the degraded fitness function of string i, $f(x_i)$ is the original fitness function of string i, $d(x_i,x_j)$ is the measure of distance between strings i and j, $share(d(x_i,x_j))$ is the sharing

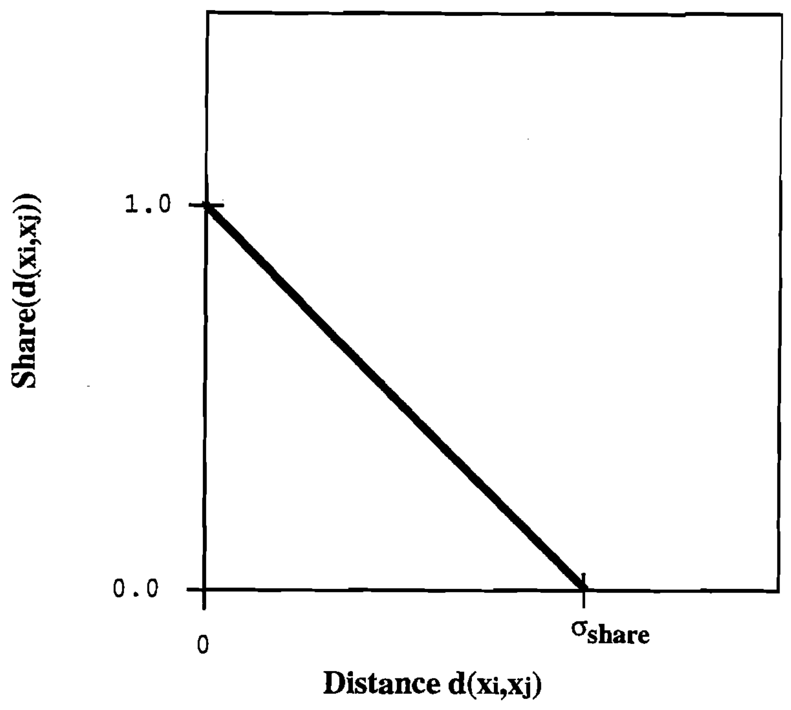

Figure 4. Triangular Sharing Function from *Goldberg and Richardson* [1987]

function contribution of string j on string i using the triangular sharing function, and N is the number of strings in the current population.

Oei et al. [1989] found that this method of sharing produces instability in the population when used in conjunction with tournament selection. In order to obtain a stable population they suggest using a method called continuously updated sharing when using tournament selection. In continuously updated sharing the current population is shuffled to produce random tournament pairings. The first tournament is held without performing a sharing function calculation and the winner is placed into the mating population. The rest of the tournaments are held one by one with only the strings already in the mating population contributing to the sharing function calculations of the strings in the other tournaments. In this way the strings participating in the first tournaments do not have their fitness function degraded by strings in the current population. However, as the mating population begins to fill up, the fitness functions of strings that are similar to those already in the mating population are degraded, while the fitness functions of dissimilar strings, as well as strings already present in the mating population, are not degraded. Continuously updated sharing specifically maintains diversity in the mating population.

For most single-objective problems the distance measure is defined in terms of decision space values. *Horn and Nafpliotis* [1993] and *Fonseca and Fleming* [1993] were able to maintain a diverse population of strings for multiple objective problems by computing the distance measure in terms of objective function values. By sharing in objective space they directly force the solutions to spread out across a wide range of objective function values. Their formulations appear to maintain a stable population of non-dominated solutions along the tradeoff curve, but neither paper compares the GA solutions to a previously determined true tradeoff curve.

4.4 Pareto Optimal Ranking with Sharing

The only difference between the basic Pareto-optimal ranking GA and the Pareto-optimal ranking with sharing is the use of a sharing function. The GA formulation presented here uses *Oei's* continuously updated sharing scheme and *Goldberg and Richardson's* triangular sharing function in conjunction with *Goldberg's* Pareto-optimal ranking scheme.

At the beginning of each generation the contaminated area and reliability score is calculated for each string in the population. The strings are then ranked according to Pareto non-dominance, and the population is shuffled to create random tournament pairings. Before the winner of each binary tournament is determined the competing strings have their ranks degraded by comparing each string to the strings already in the mating population. The following sharing function is used

$$\mathrm{Rank}_i^{new} = \mathrm{Rank}_i^{old} + \mathrm{share}_i \tag{13}$$

where,

$$\mathrm{share}_i = \sum_{j=1}^{N} \frac{\sigma_{share} - d_j}{\sigma_{share}} \tag{14}$$

and

$$d_j = \begin{cases} |\,\mathrm{Reliability}_i - \mathrm{Reliability}_j\,|, & \text{if } |\,\mathrm{Reliability}_i - \mathrm{Reliability}_j\,| \le \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Reliability is given as a percentage of the simulated plume realizations that are detected by each monitoring well system.

Note that the sharing function is based solely on the reliability objective and does not include the contaminated area objective. This is because our intent is to find the tradeoff curve along the entire range of the reliability objective. By sharing in only the reliability objective we can obtain this diversity and then allow the Pareto-optimal ranking selection scheme to improve the contaminated area objective so that the population stays close to the tradeoff curve while being dispersed along the entire curve.

4.5 Maintaining the Pareto Optimal Set

The GA search method tends to converge to a single Pareto-optimal solution after a large number of generations (in our particular case full convergence has taken place after approximately 70 generations). At this point the random mutation is the only source left to maintain diversity in the population, and the power of the GA to find new Pareto-optimal points on the tradeoff curve is greatly diminished since a diverse population of "good" building blocks is essential for GAs to work properly. A high mutation rate can maintain a diverse population, but does not insure that the required "good" building blocks will be maintained. For this reason "Pareto-optimal re-injection" is used to maintain a diverse set of "good" building blocks.

In order to incorporate Pareto-optimal re-injection the GA searches through the current population at the end of each generation. All Pareto-otimal solution strings are extracted from the population and recorded to a file. These strings are then compared to the Pareto-optimal strings of previous generations and the non-dominated strings are recorded in a best-of-run file. These best-of-run strings are re-injected back into the current population by randomly replacing strings in the current population. In this way all Pareto-optimal strings previously found in any generation are now present in the current population. Pareto-opitmal re-injection allows the GA population to maintain a set of the best building blocks. This is something that sharing alone cannot guarantee since crossover between Pareto-optimal strings may result in a degradation of both solutions and a loss of the best building blocks.

At the end of a GA iteration, consisting of a fixed number of generations with constant parameter values, the final population is sorted and all non-dominated strings are combined to form a best-of-run set which comprises the tradeoff curve for that iteration.

Chapter 5

GA Coding and Formulations for the Monitoring Problem

5.1 Coding of Decision Set Strings

The first step in formulating a GA is to determine how to code the GA decision set (coding, here, refers to determining how a given decision set of active wells is to be represented as a GA string). For coding convenience we hold the number of active wells constant, so that we can generate tradeoff curves of reliability vs. contaminated area. Under this coding each potential well location is represented by a X-Y integer pair corresponding to its row and column location in the set of potential monitoring wells. The length of a string is two times the number of active wells in the monitoring system. An integer, corresponding to a column or row number in the potential well field, is placed in each of the string's alleles. Integers between one and the number of rows of potential monitoring wells are placed in each string's x-positions and integers between one and the number of columns of potential monitoring wells are placed in each string's y-positions. Each string now represents a system of monitoring wells which map to their real-world potential well locations as shown in Figure 5. Several runs can be made varying the length of the string to find the three-way tradeoff of reliability vs. contaminated area vs. number of wells.

5.2 Selection of Parameter Values

Performance of the formulations above relies on proper setting of the P_{cross} and P_{mute} parameters. Initial test runs indicate that values of P_{cross} in the range of 0.70-0.90 produce the best results, and that the performance of the GA is robust across this range of P_{cross} values. For all of our runs we will be using a P_{cross} value of 0.80. This P_{cross} value means that, on average, one of every five strings that survive into the mating population will not participate in the mating operation. Instead these strings will survive into the next generation.

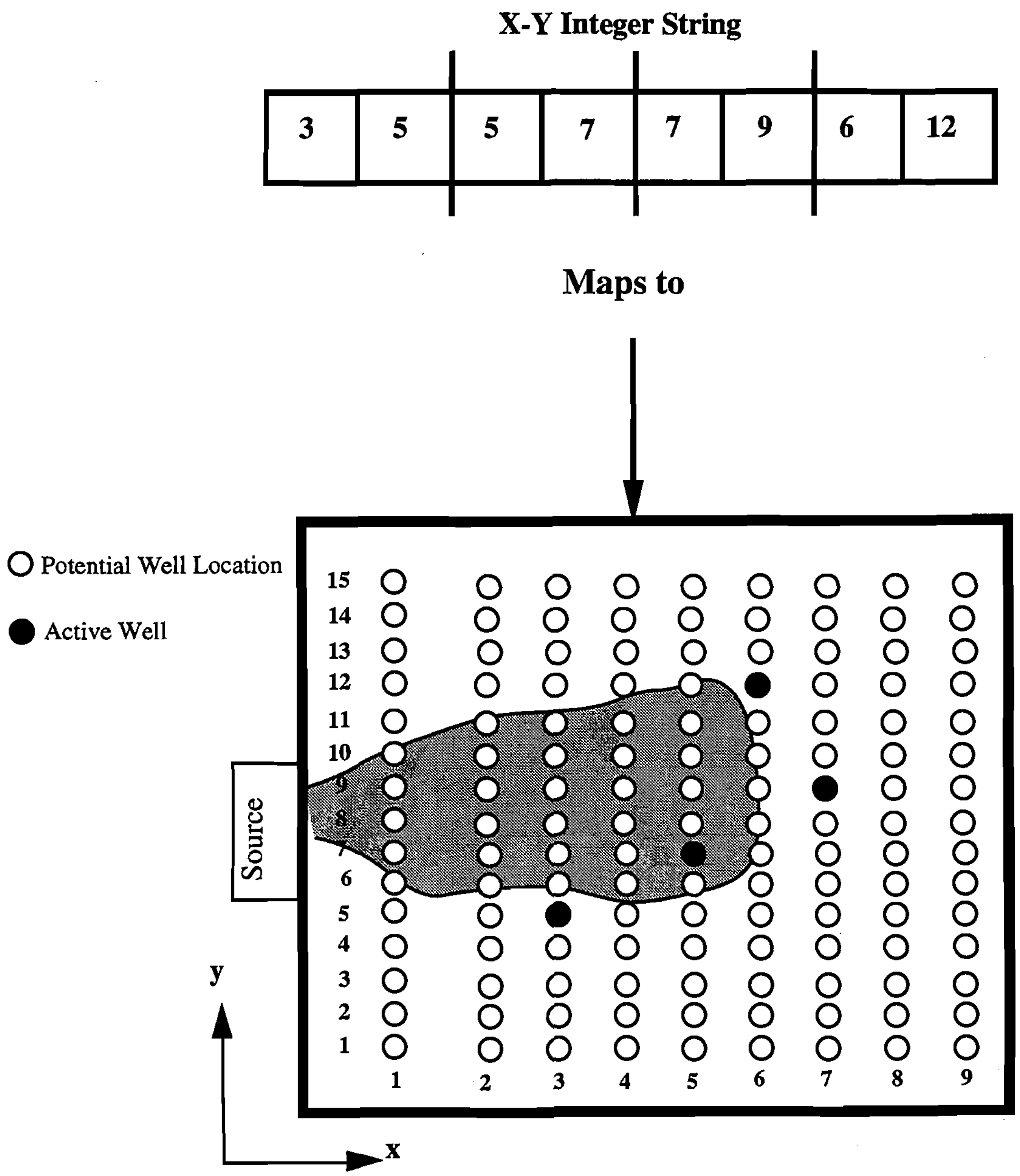

Figure 5. Mapping of GA String to Real-World Locations

When selecting a P_{mute} value the goal is to maintain genetic diversity while attempting to avoid making the mutation operator the primary search technique. Initial runs showed that best results were obtained when, on average, one in every four to one in every eight strings is mutated each generation. This means that for a string of length five P_{mute} would range between 0.05 and 0.025. Performance of our GA is also robust within this range of values. A value of P_{mute}=0.2/(string length) will be used for all runs (i.e., for a string of length five $P_{mute} = 0.04$), resulting in an average of one out of every five strings being mutated each generation.

When performing the mutation the GA steps through each allele on every string. A random number is drawn from a uniform distribution between zero and one, and if it is less than P_{mute} the allele is mutated. When mutation occurs the mutated allele is replaced by a random integer between one and the number of monitoring well rows is selected if the allele is an x-allele is selected. If the allele is a y-allele then the GA randomly selects an integer between one and the number of monitoring well columns to replace the old allele value. Each allele is mutated independently from all other alleles. Therefore, it is possible for two or more alleles on the same string to be mutated in the same generation.

When incorporating sharing into the Pareto-optimal ranking GA the value of σ_{share} must be carefully chosen. Too small a value of σ_{share} diminishes the effects of sharing, because not enough strings will contribute to each other's sharing function calculations. Too high a value for σ_{share} will cause too many strings to contribute to the sharing function calculation and the GA population will become chaotic. Values of σ_{share}= 2.5%, 1.0%, and 0.5% are investigated to determine the optimal setting of σ_{share} for this particular problem.

5.3 Genetic Algorithm Formulations

Using the coding, crossover, and selection schemes discussed above, five genetic algorithm formulations were developed to attempt to find the two-way tradeoff of reliability vs. contaminated area. The only difference between the formulations is in the method that each formulation uses to determine a string's fitness value. All five of the GA formulations follow the basic procedure shown in Figure 6. A generation is defined as one trip through the flowchart loop, while completion of the entire flowchart constitutes an iteration. The first GA formulation is a reference formulation to test the power of the GA and to record a performance baseline by which to evaluate the other four formulations.

5.3.1 Weighted Objective Function

The first step in the investigation is to determine if genetic algorithms have the problem solving power required to solve the EPMP. This is done by duplicating the results *Meyer et al*. [1992] obtain through the use of simulated annealing using equation (10). Reliability is measured in the number of realizations whose plumes are detected divided by the total number of simulated plumes. Contaminated area is calculated by summing the normalized contaminated areas of all detected plumes. The GA is run several times changing β to find the tradeoff curves for Q=5.

This formulation is run a second time using *Steuer's* [1986] Tchebycheff weighting method function as its sole fitness function. The weighting factor, β, is allowed to vary from 0.0 to 1.0 with uniform steps of 0.05. (Tchebycheff weighting factors are limited to a range from 0.0 to 1.0).

5.3.2 Multi-Objective GA Formulations

The four multi-objective GA formulations discussed in Chapter 4 will be tested for their ability to generate the multi-objective tradeoff curve for two example data sets for the EPMP groundwater monitoring problem. The four formulations: Vector Evaluated GA, Pareto-optimal Ranking GA, Combination Vector Evaluated GA and Pareto-optimal Ranking GA, and the Pareto-optimal Ranking GA with Sharing will be referred to as the VEGA, Ranking, Combination, and Ranking with Sharing formulations, respectively, in the remainder of this paper.

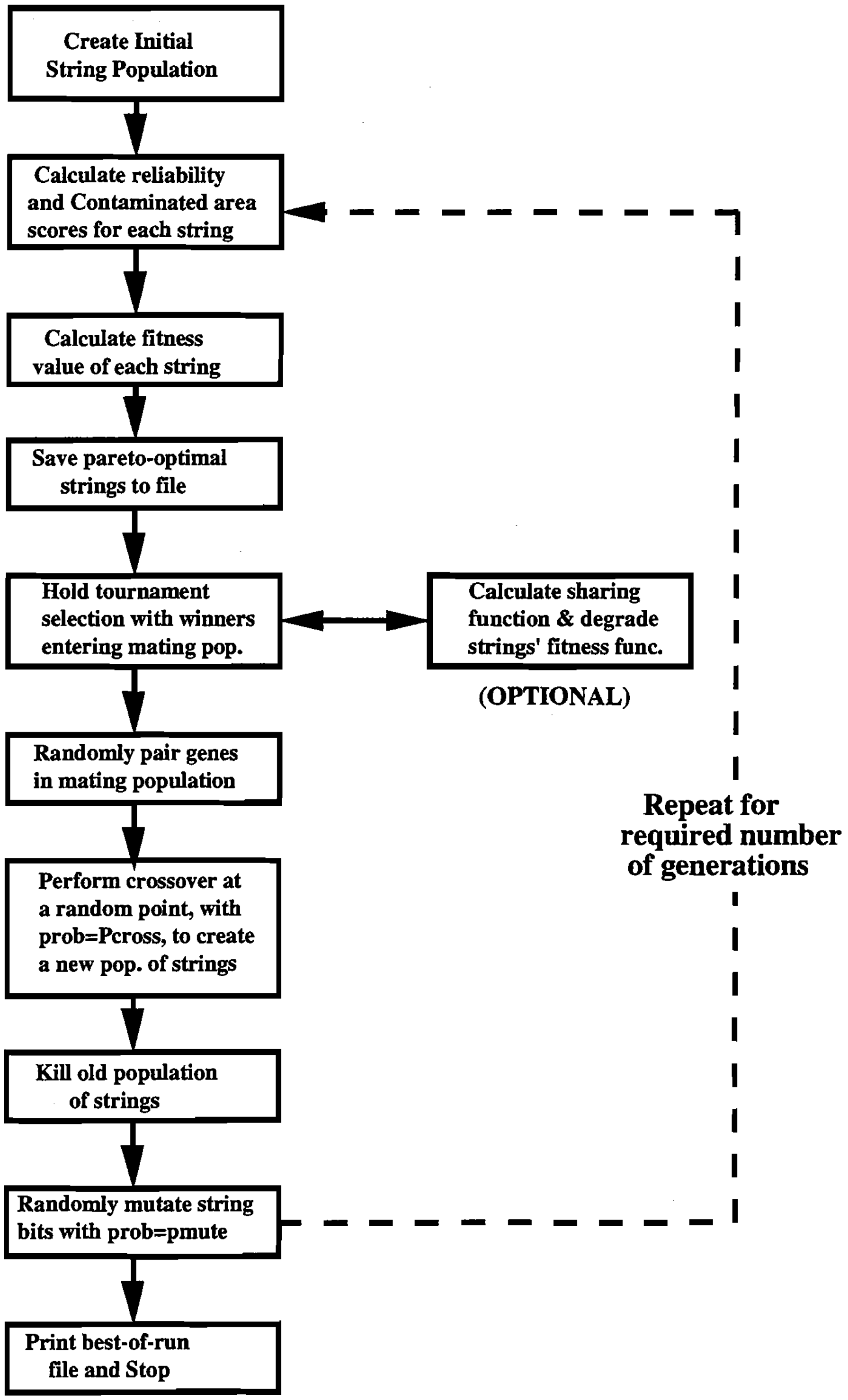
Create Initial String Population
Calculate reliability and Contaminated area scores for each string
Calculate fitness value of each string
Save pareto-optimal strings to file
Hold tournament selection with winners entering mating pop.
Calculate sharing function & degrade strings' fitness func.
(OPTIONAL)
Randomly pair genes in mating population
Perform crossover at a random point, with prob=Pcross, to create a new pop. of strings
Kill old population of strings
Randomly mutate string bits with prob=pmute
Print best-of-run file and Stop
Repeat for required number of generations

Figure 6. GA Flow Diagram

Chapter 6

Results

For the five GA formulations discussed in Chapter 5, two separate data sets are generated. The first set consisting of 200 simulated plume realizations and 135 potential well locations is thoroughly investigated. Several different values for each of the GA parameters are tested in order to determine the optimal setting for these parameters. The first data set is thoroughly investigated to insure that the GAs are working properly and that the correct setting for each parameter is chosen. This investigation focuses on finding a tradeoff curve of reliability vs. contaminated area for the case where Q = 5. This case is more difficult than lower values of Q, and provides a good baseline by which to judge the GA's performance.

The second data set corresponds to a more difficult problem consisting of 1000 simulated plume realizations and 470 potential well locations for Q = 10. This problem is much more computationally intensive, in terms of CPU, so the parameter settings are not investigated quite as thoroughly as for the first data set. Instead the best parameter settings determined from the first data set are applied to this more difficult problem. This allows us to determine if parameter settings can be generalized from the simpler problem to the more difficult one.

The modeled data sets are from the work of *Meyer et al.* [1992] and *Meyer* [1992] which incorporate uncertainty by allowing for spatial variability of hydraulic conductivity and random leak locations. The hydraulic conductivity fields are generated using TUBA (*Zimmerman and Wilson* [1990]) which incorporates the Turning Bands method of *Mantoglou and Wilson* [1982]. Hydraulic conductivity is a log-normally distributed parameter with a covariance that exponentially decreases with separation distance, as suggested by *Freeze* [1975]. Leak location is randomly drawn from a uniform distribution over the down gradient boundary of the landfill, and *Meyer* [1992] model groundwater solute transport using *Sudicky's* [1989] Laplace Transform Galerkin computer code. Table 1 lists the significant hydraulic parameter values used for simulating the plume realizations for the data sets. *Meyer et al.* [1992] found that the reliability

Parameter	Data Set #1	Data Set #2
Number Pot. Mon. Wells	135	470
Number of Pot Mon. Well Rows	9	10
Number of Pot. Mon. Wells Per Row	15	47
X-dimension	2000 m	400 m
Y-dimension	1000 m	250 m
X-spacing of wells	150 m	2 - 10 m
Y-spacing of wells	25 m	2.5 m
Source length	100 m	100 m
Source concentration	1.0 mass units/m^3	1.0 mass units/m^3
Number of Source nodes	41	41
Zero flux boundaries	y = 0 m and 1000m	y = 0 m and 250 m
Avg. gradient in x-direction	0.00167	0.00167
Source flowrate	0.0018 m^3/day	0.0018 m^3/day
Detection limit	0.005 mass units/m^3	0.005 mass units/m^3
Diffusion coefficient	2.16 x 10^(-5) m^2/day	2.16 x 10^(-5) m^2/day
Mean of ln(Ks)	0.79	0.79
Variance of ln(Ks)	0.96	0.96
Correlation Scale of ln(Ks)	20.0 m	20.0 m
Longitudinal Dispersivity	2.0 m	2.0 m
Transverse Dispersivity	0.2 m	0.2m

Table 1. Hydraulic Parameters for Groundwater Flow Simulation

and contaminated area estimates are sensitive to the number of plume realizations and the assumed probability distribution for hydraulic conductivities.

These data sets are used for each of the formulations to allow for comparisons between the methods. Using these data sets, all GAs and the simulated annealing algorithm were run on the same model of HP-UX workstation in order that results and computational time could be properly compared. GA parameter settings for the two data sets are listed in Table 2.

6.1 First Data Set: 200 Plumes, 135 Potential Well Locations

6.1.1 Weighted Objective Function

The weighted GA tradeoff curve corresponds exactly to that of the simulated annealing procedure developed by *Meyer et al.* [1992]. The results are shown in Figure 7. Only the points on the tradeoff curves are obtainable solutions. The lines connecting these points are included only to show the general trend of the curves. These curves are referred to as the "true"

Parameter	**Data Set #1**	**Data Set #2**
Number Pot. Mon. Wells	135	470
Number of Pot Mon. Well Rows	9	10
Number of Pot. Mon. Wells Per Row	15	47
Pcross	0.80	0.80
Pmute	0.04	0.04
Sigma Share	0.5%,1.0%,and 2.5%	1.0%
Nchange / Prel (Combination)	30,50,and 70 / 0.50,0.70,and 0.90	30 / 0.90
Prel (VEGA)	0.50,0.70,and 0.90	0.70

Table 2. GA Parameters

tradeoff curves in the remainder of this paper, and will be shown as dashed lines connecting large open points in all other graphs. Because the weighting method misses non-convex portions of the tradeoff curve and because neither GAs nor simulated annealing guarantees the theoretical maximum these curves are not necessarily the true tradeoff curve. However, the fact that both method find the same points provides some evidence that the obtained points are Pareto-optimal. The results of the other four genetic algorithm formulations are compared to these results.

Using *Steuer's* [1986] Tchebycheff method to obtain the non-convex portions of the tradeoff curve produced some disappointing results. Steuer's method produced only the two different Pareto-optimal points (both already found using the linear weighting method) over the range of weighting factors discussed in Section 4.4.1. These results will be re-examined later in this section.

6.1.2 Vector Evaluated GA

Three separate runs were made using the VEGA formulation to investigate the effect of changing P_{rel}, the probability of selecting the binary tournament winner based on reliability, on the performance of the GA. The settings of this parameter are; 0.50, 0.70, and 0.90 (corresponding to probabilities of selecting for lowest contaminated area of 0.50, 0.30, and 0.10, respectively). A probability of selecting for reliability of 0.70 means that, on average, 70% of the selections will pick the tournament winner on the basis of a higher reliability score, while the

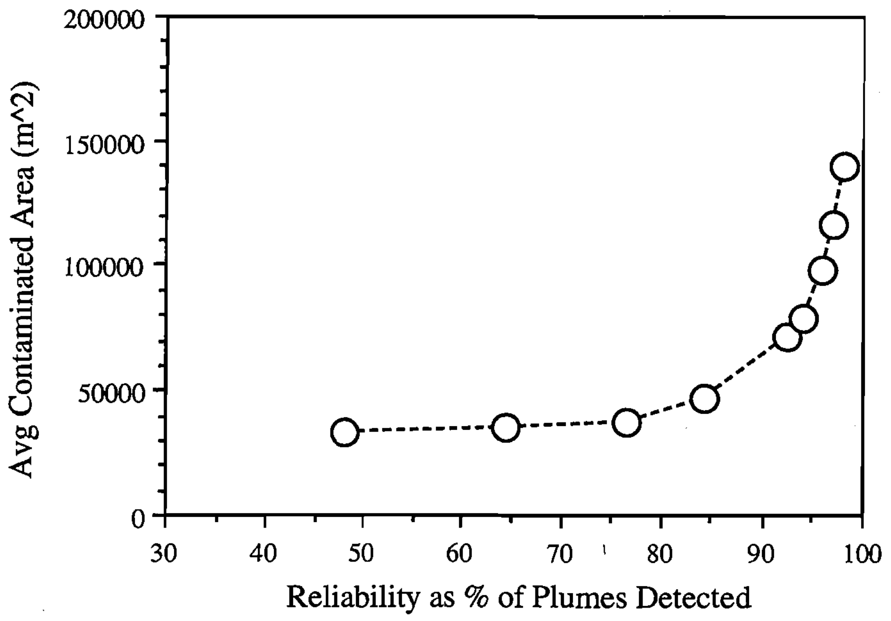

Figure 7. "True" Tradeoff Curve (Q = 5, NPOTW=135)

other 30% of the selections will choose the winner based on lower average contaminated area. The results of the P_{rel}= 0.50 and 0.70 are shown in Figure 8 (P_{rel} = 0.90 is not shown because the results are poor and including the results makes the graph difficult to interpret).

It is difficult to determine which setting for P_{rel} is best. A setting of 0.50 is best for finding the lower reliability end of the tradeoff curve, but suffers somewhat near the high reliability end of the curve (performance is evaluated according to the degree to which the GA finds the actual points on the Pareto-optimal frontier). A setting of 0.70 leads to better performance at the upper and middle reliability portions of the tradeoff curve, but this gain is offset by poorer performance at the low reliability end. Finally, a setting of P_{rel} = 0.90 leads to poor performance everywhere except the highest reliability end of the tradeoff curve. This is most likely due to the fact that at this high P_{rel} value, contaminated area plays an insignificant role in determining the survival of a string.

Intuitively we would think that a higher value of P_{rel} will cause the population to focus at a higher reliability solution, and this may be the reason that different settings of P_{rel} perform better at different portions of the tradeoff curve. As the population focuses on a particular

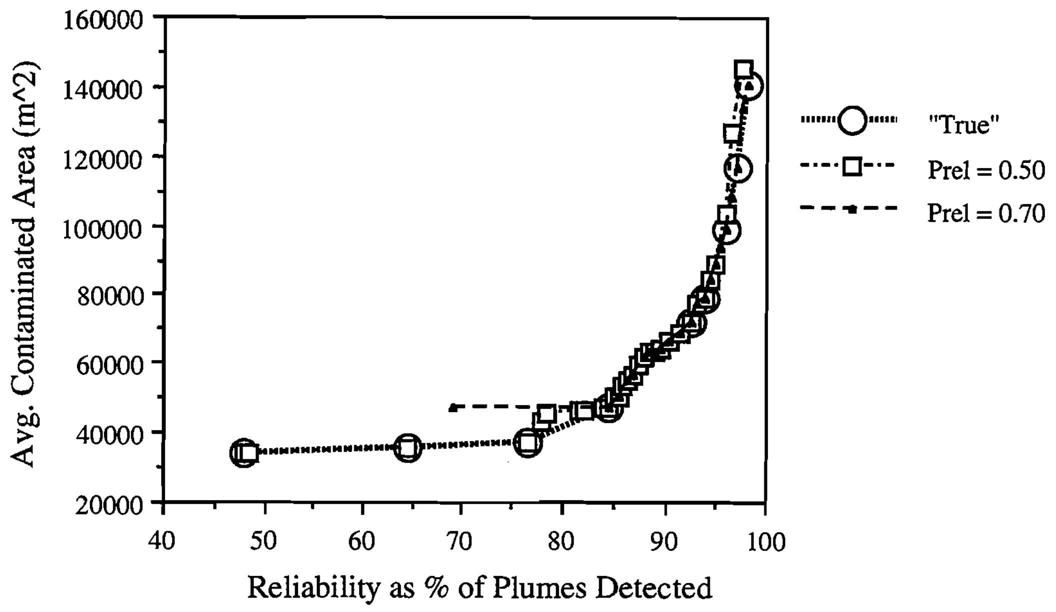

Figure 8. VEGA vs. "True" (Q=5, NPOTW=135)

solution the GA continues to search for solutions near the focal point in objective space, thus doing a good job of generating the tradeoff curve closest to the focal point of the VEGA search.

6.1.3 Pareto-Optimal Ranking

The Pareto-optimal ranking formulation uses a fitness value that is quite different from the one used in the VEGA formulation and the final results appear better than those obtained using the VEGA formulation (see Figure 9). The Pareto-optimal ranking scheme is able to generate all but the very highest reliability end of the tradeoff curve. This may be because no significant genetic pressure for improvement exists to force the solution towards the highest reliability end of the tradeoff curve.

6.1.4 Combination of VEGA and Ranking

The combination VEGA and Pareto-optimal ranking formulation evolved because of the inability of either VEGA or Pareto-optimal ranking to find the entire tradeoff curve, especially near the high reliability end. By combining the two schemes the population may be forced to

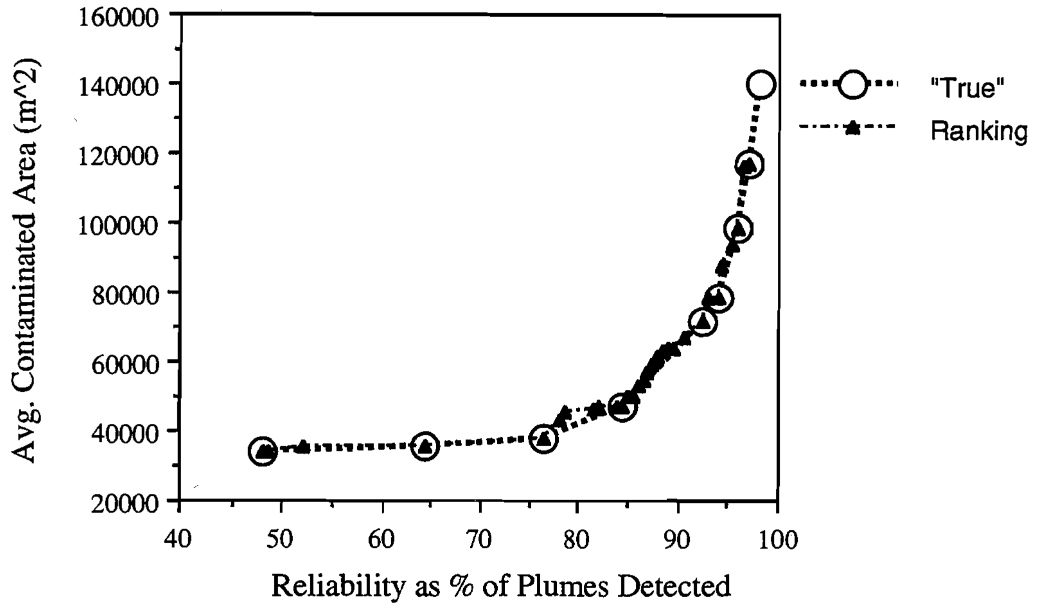

Figure 9. Ranking vs."True" (Q=5, NPOTW=135)

the high reliability end of the tradeoff curve in the early generations, using the VEGA formulation with a high value for P_{rel}, and in later generations the population is allowed to drift back towards the knee and lower reliability end of the tradeoff curve, using the Pareto-optimal ranking scheme.

Values of 30, 50, and 70 are used for N_{change}, and 0.50, 0.70, 0.90 are used for P_{rel}. All combinations of these parameters are investigated for Q = 5. Best results were obtained for N_{change}=30 and P_{rel} =0.90. These results are compared to the results of N_{change}=50 and P_{rel}=0.70 in Figure 10.

Once again the GA, after adjustment of the parameters, performs well in finding the lower and middle reliability portions of the tradeoff curve. Performance suffers only at the very highest reliability portions of the curve. When P_{rel} is increased, N_{change} must be reduced or else performance begins to suffer at the low reliability end of the tradeoff curve.

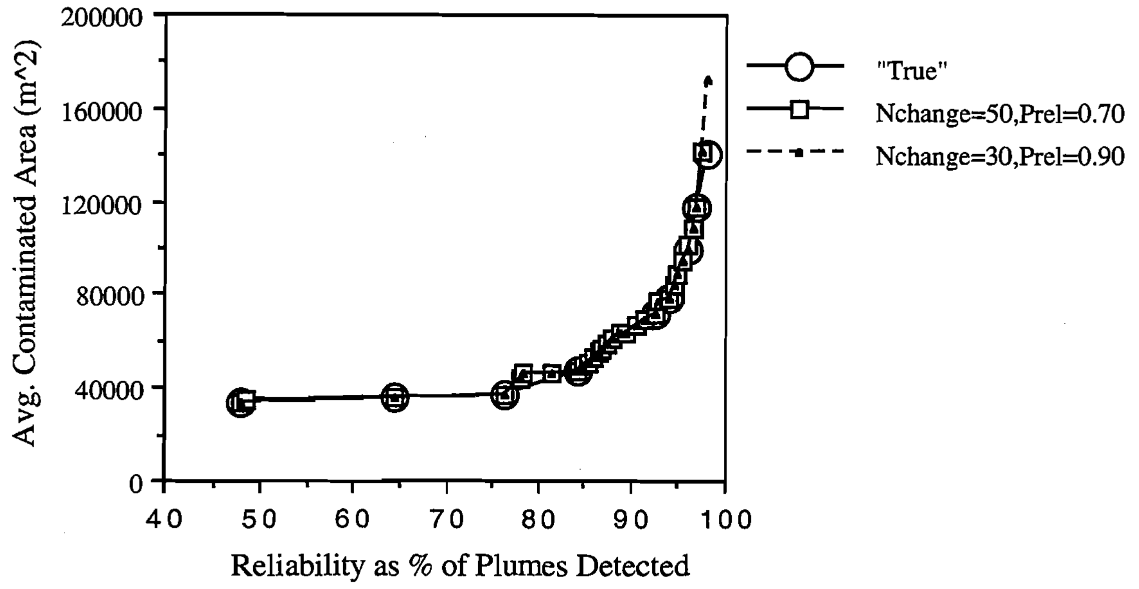

Figure 10. Combination vs. "True" (Q=5, NPOTW=135)

6.1.5 Pareto-Optimal Ranking with Sharing

Each of the σ_{share} values are run for two different random number seeds. Figure 11 shows a comparison of the best tradeoff curve generated by each σ_{share} value for the EPMP and the "true" tradeoff curve. A value of $\sigma_{share} = 1.0\%$ of the total plume realizations outperform $\sigma_{share} = 2.5\%$ and 0.5% (which is not shown in Figure 11 for purposes of clarity) in terms of generating the tradeoff curve.

A value of $\sigma_{share} = 0.5\%$ is too small and defeats the purpose of sharing by severely limiting the number of strings that contribute to the sharing function of other strings. Too high a setting degrades performance by allowing contribution by too many adjoining strings. A value of $\sigma_{share} = 1.0\%$ seems to allow the sharing process to work most effectively.

6.2 Comparison of Results

6.2.1 Genetic Algorithm Formulations

All four of the multiple-objective GA formulations were able to find large portions of the tradeoff curve of reliability vs. contaminated area, although none were able to find the entire

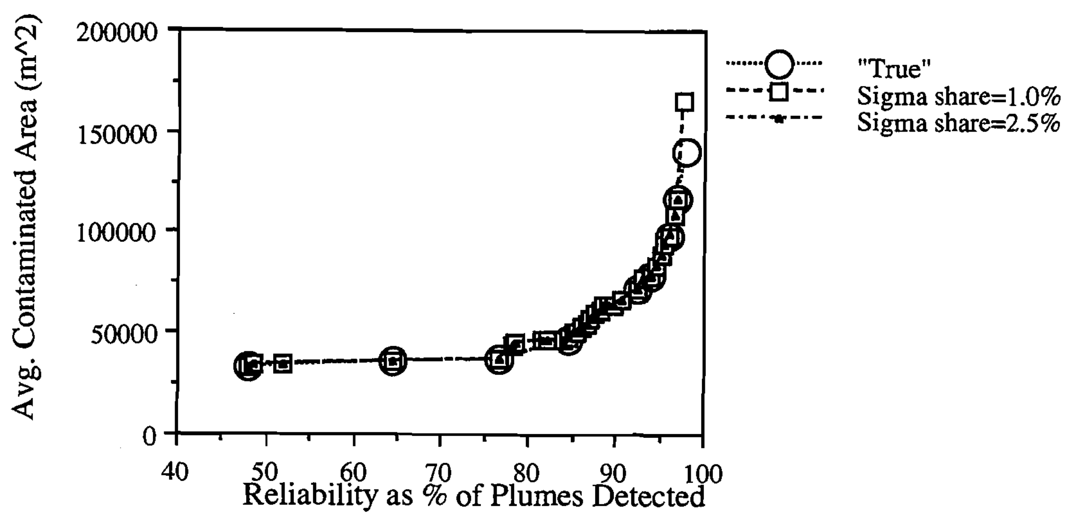

Figure 11. Ranking w/ Sharing vs. "True" (Q=5, NPOTW=135)

tradeoff curve. All four formulations lack the ability to find both ends of the tradeoff curve simultaneously. The combination formulation performs better than any of the others. Figure 12 shows a comparison of the tradeoff curve generated by each of the four formulations: Pareto-optimal ranking, Combination (P_{rel}=0.90,N_{change}=30), VEGA (P_{rel}=0.70), and Pareto-optimal ranking with sharing (σ_{share}=1.0%). As performance increases at one end of the curve it suffers at the other end, although the combination and the ranking with sharing formulations come very close to finding the entire tradeoff curve. However, in many cases the decision maker is interested in a specific region of the tradeoff curve (e.g., in our case we are particularly interested in the high reliability end). If this is true, both the VEGA and the combination formulations allow us to specify P_{rel} in order to force the population to the portion of the curve that the decision maker is most interested in.

The VEGA, combination, and ranking with sharing formulations require the adjustment of additional parameters (σshare, P_{rel} and N_{change}). The addition of these parameters may require additional GA iterations in order to find the best settings for each particular problem.

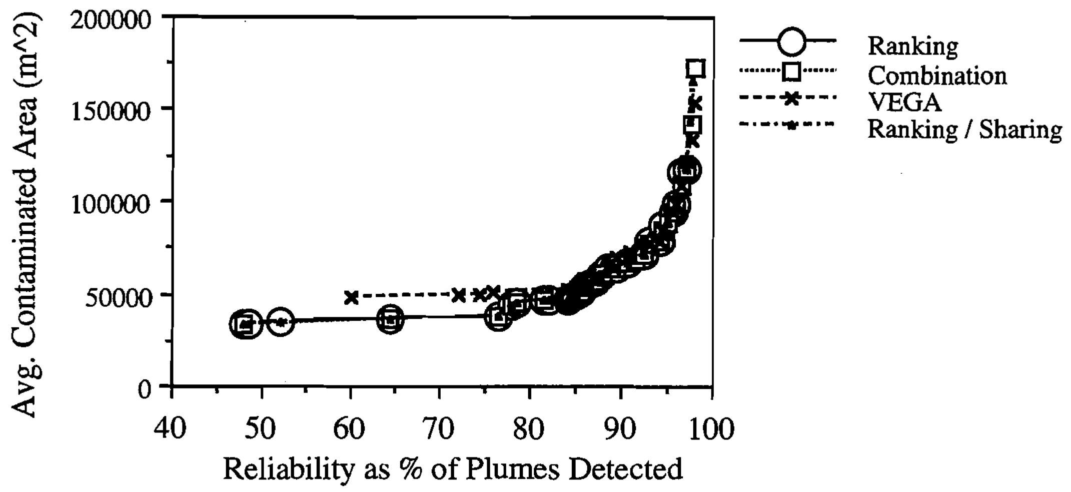

Figure 12. Comparison of Multi-Objective GA Formulations

Workstation run time for all four formulations were similar in magnitude. Average run times for 1000 strings and 100 generations were approximately two to three minutes for the VEGA, ranking, ranking with sharing, and combination formulations (See Table 3 for comparison of CPU requirements). Note that the 100,000 objective function evaluations of the multiple objective GA formulations represent only a fraction of the 3.47×10^8 possible well combinations for Q=5. The Pareto-optimal ranking with sharing formulation requires the most computation of any of the multiple-objective GAs. This is because the population must be sorted and ranked and the sharing calculations must be performed during each generation. Approximately 60-70% of the computational time is spent doing objective function evaluations. Since this operation is highly parallelizable time-savings could be realized through use of parallel computers.

Formulation	CPU, units	Pop. Size	# Generations	# Obj. Func. Eval.
VEGA	130.5	1000	100	100,000
Ranking without Sharing	200.5	1000	100	100,000
Ranking with Sharing	245.7	1000	100	100,000
Combination	164.2	1000	100	100,000
Weighted GA	923.4	750	100	900,000
Simulated Annealing	242.9	---	---	71,433

Table 3. Comparison of Computational Requirements

6.2.2 Ranking with Sharing vs. Ranking

The "true" tradeoff curve is compared to the curves generated using the sharing GA (σ_{share} = 1.0%) and the simple Pareto-optimal ranking GA in Figure 13. Neither of the multi-objective GAs are able to generate the highest reliability end of the tradeoff curve, and the sharing GA only marginally outperforms the simple Pareto-optimal ranking GA. Since the sharing GA is 20% more computationally intensive, in terms of CPU, than the simple Pareto-optimal GA, why should one consider using the sharing GA?

The answer becomes clear when examining the final population of solutions for the sharing GA and the simple Pareto-optimal ranking GA. The distribution of final populations in objective space is shown in Figures 14 and 15 for the GAs with and without sharing. Although the figures make it appear that the sharing GA has a larger population this is not the case. Both GAs have a population of 2000 strings, but the GA without sharing has more identical solutions in the final population than the sharing GA. Therefore, many of the points plotted in Figure 15 lie directly on top of one another.

In order to compare the diversity of the final populations objectively the final populations of the Pareto-optimal ranking and the Pareto-optimal ranking with sharing formulations were evaluated using the following diversity measure.

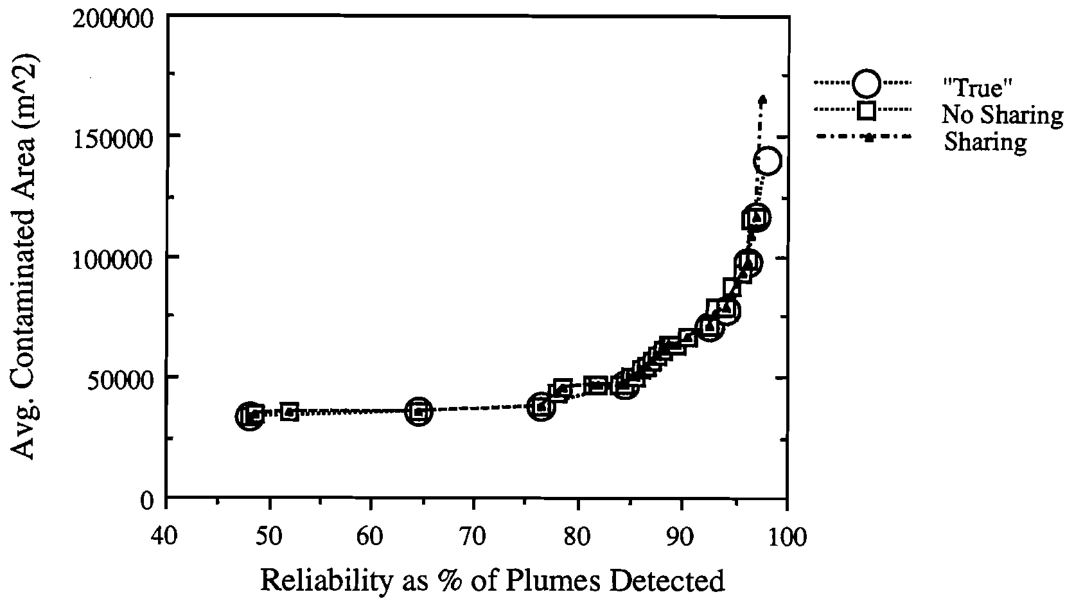

Figure 13. Comparison of Sharing vs. No Sharing

$$\text{Diversity} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left| \text{Reliability}_i - \text{Reliability}_j \right| \qquad (16)$$

where N is the number of strings in the population. Diversity scores for the formulations are 32,129,386 and 136,010,362 for the Pareto-optimal ranking and Pareto-optimal ranking with sharing formulations, respectively. The diversity measure shows that the final population of the Pareto-optimal ranking with sharing formulation is much more diverse than the formulation without sharing. What exactly does this diversity obtain us if the tradeoff curves of the two GAs are so similar?

Both the sharing GA (σ_{share} = 1.0%) and the GA without sharing were run for five different random number seeds. While the sharing GA consistently generated approximately the same tradeoff curve all five runs, the GA without sharing only generated its best tradeoff curve in two out of the five trials. This suggests that the diversity helps maintain consistency in generating the tradeoff curve, probably because a diverse population is so essential for a GA to find new solutions.

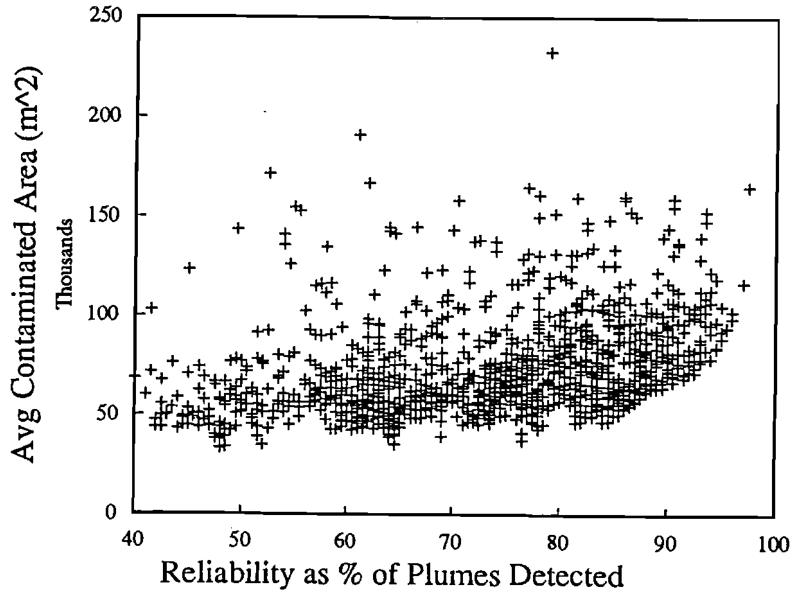

Figure 14. Distribution of Generation 200, Ranking with Sharing

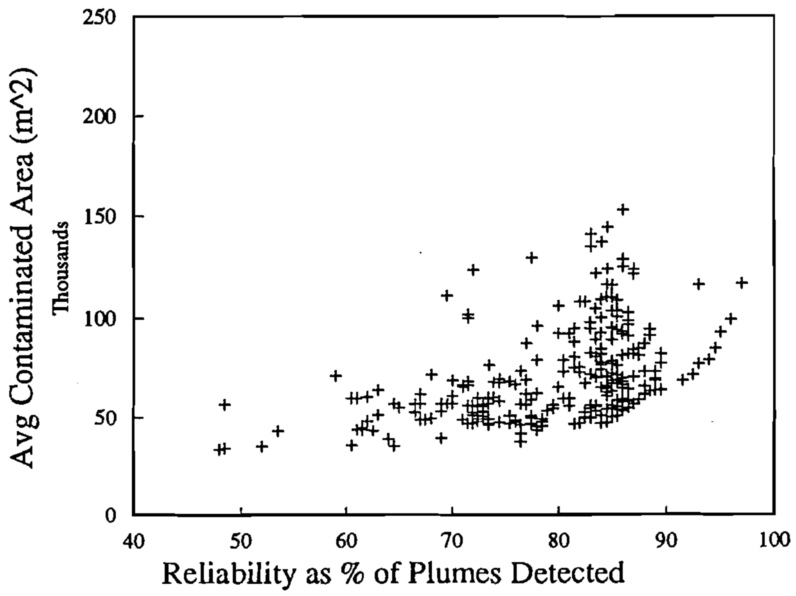

Figure 15. Distribution of Generation 200, Ranking without Sharing

6.2.3 Genetic Algorithms vs. Simulated Annealing

Genetic algorithms have a distinct advantage over simulated annealing in that they are able to consider both objective values simultaneously. The simulated annealing algorithm requires a single weighted objective function. This necessitates running the simulated annealing algorithm several times with different weights to find the tradeoff curve. The Pareto-optimal ranking tradeoff curve (and also the VEGA and combination curves) captures these non-convexities of the tradeoff curve and is, therefore, more accurate in this region than the "true" tradeoff curve.

Simulated annealing does have a computation time advantage over the genetic algorithm. Each simulated annealing iteration requires approximately 10 to 15 seconds run time on the HP-UX workstation to converge to an optimal solution. Therefore, approximately 10-12 different weights can be tried in the time it takes the multiple-objective GAs to complete a single iteration. However, simulated annealing finds only a single Pareto-optimal point each iteration while GAs find a large number of Pareto-optimal points each iteration. Therefore, the multi-objective GAs are just as efficient, in terms of computer time, in generating the tradeoff curve of reliability versus contaminated area. CPU times for Q = 5 are shown for simulated annealing, and the four GA formulations in Table 3. For simulated annealing and the weighted GA formulation the run time constitutes the time it takes to run 12 different weighting factors in series (i.e., 12 iterations).

The sharing GA also has an advantage over simulated annealing because the final population contains a large number of alternatives that are near the tradeoff curve, but different from each other in decision space. This diversity of solutions allows the decison maker to evaluate several different alternative that are similar to each other in objective space but very different in decision space. Simulated annealing converges to a single, optimal solution rather than a population of solutions. Therefore, simulated annealing does not find alternatives that are near the tradeoff curve and different form each other in decision space.

The speed of the GA can be increased by parallelizing the fitness function evaluations which are independent of each other. This would decrease the GA run time by approximately 60-70%. It should be noted that simulated annealing with a weighted objective function could also be parallelized by running different weighting function values in parallel, but the simulated annealing search method for a particular weighting factor values is inherently serial.

The weighted GA performs identically to simulated annealing in terms of results, but simulated annealing is approximately four times faster than the weighted GA. Both methods were able to find the tradeoff curves, except where they miss non-convexities.

6.3 The Tchebycheff Method Revisited

After obtaining the results from all four of the multi-objective GA formulations a data set was compiled of all Pareto-optimal points found by any of the GA formulations for $Q = 5$. These results were examined to determine the values of β required in the Tchebycheff method in order to obtain all of the Pareto-optimal points found by any of the GA formulations. The Tchebycheff weighted GA was then run with each of these weighting factors to determine if the multi-objective GA's were indeed finding non-convexities of the tradeoff curve and not simply sub-optimal points.

These results, shown graphically in Figure 16, indicate that the multi-objective GAs are finding non-convexities of the tradeoff curve since the results match those to the Tchebycheff method. Some of the β values required to find all of these non-convex points must be extended to the fifth decimal place. In order to find all of Pareto-optimal points using the Tchebycheff method, without a priori knowledge of where these points were, would require an interactive trial-and-error method of trying a β value, studying the results, and then trying new values of β. The GA based method allows us to avoid this iterative type approach.

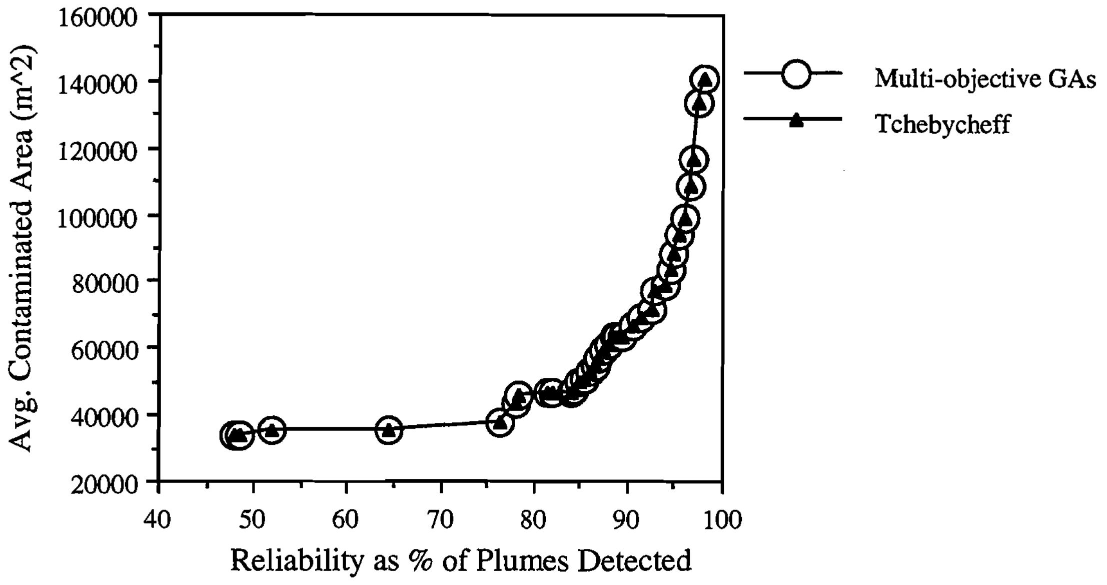

Figure 16. Tchebycheff Method vs. Multi-Objective GAs

6.4 Second Data Set: 1000 Plumes, 470 Potential Well Locations

This data set represents a much more difficult problem than the first data set. Choosing 10 active wells from a system of 470 potential monitoring wells yields 1.31×10^{20} possible combinations of active wells as compared to 3.47×10^{8} possible combinations of 5 active wells out of 135 potential wells. For this reason the parameter setting response sensitivity is not investigated as thoroughly as for the first data set. A population size of 5000 strings will be used for each of the GA formulations due to the increased complexity of the problem. Since population size affects convergence time the GAs will be run for 200 generations.

Also, the "true" tradeoff curve is obtained by using simulated annealing and the weighted objective function of *Meyer et al.* [1992] for 14 different weighting factors ranging from 0.02 to 20.0. This "true" tradeoff curve is compared to the four different multi-objective GA formulations: VEGA (P_{rel} = 0.5 and 0.7), Ranking, Ranking with Sharing (σ_{share} = 1.0%), and Combination (N_{change}=30 and P_{rel}=0.9, and N_{change}=50 and P_{rel}=0.9). The "true" tradeoff curve suffers from the same limitation as the "true" curve for the previous data set; it does not find all of the convex and non-convex, non-dominated points of the actual tradeoff curve. It does, however, provide a good baseline by which to measure the performance of the GAs.

6.4.1 Performance of Multi-Objective GAs

Figures 17 through 22 compare the performance of the multi-objective GA formulations to the "true" tradeoff curve. Examination of each of the graphs shows that none of the formulations is able to generate the entire tradeoff curve for this problem. This result is to be expected since the GAs were not able to generate the entire tradeoff curve for the simpler problem either. However, all of the GA formulations, with the exception of the VEGA formulation with P_{rel}=0.7, were able to generate large portions of the tradeoff curve and give a very good estimation of the shape of the "true" tradeoff curve.

The GAs' performance suffers only at the very highest and very lowest reliability ends of the tradeoff curve. It is at the ends that the largest degradation of one objective must be accepted in order to obtain a small improvement in the other objective. Often times when decision makers are seeking a compromise solution, they are not even interested in the points at the very ends of the tradeoff curve since other objectives would suffer so greatly.

Comparing the performance of the GA formulations to each other, it seems that the Ranking and Ranking with Sharing formulations outperform the other formulations. The VEGA and Combination formulations suffer quite a bit at one end of the tradeoff curve or the other. The two Ranking formulations perform better than the VEGA at both ends of the tradeoff curve. Ranking with sharing outperforms the Ranking formulation at both ends.

The CPU requirement for the GAs for this problem were in the range of 7800-9200 CPU units, with the VEGA formulations having the lowest requirements and the Ranking with Sharing Formulation requiring the most CPU. This is approximately a 40-fold increase in CPU requirements compared to the previous example problem. The increase in CPU requirements is due to the increase in population size, the increase number of generations, and the increase in the number of simulated plumes.

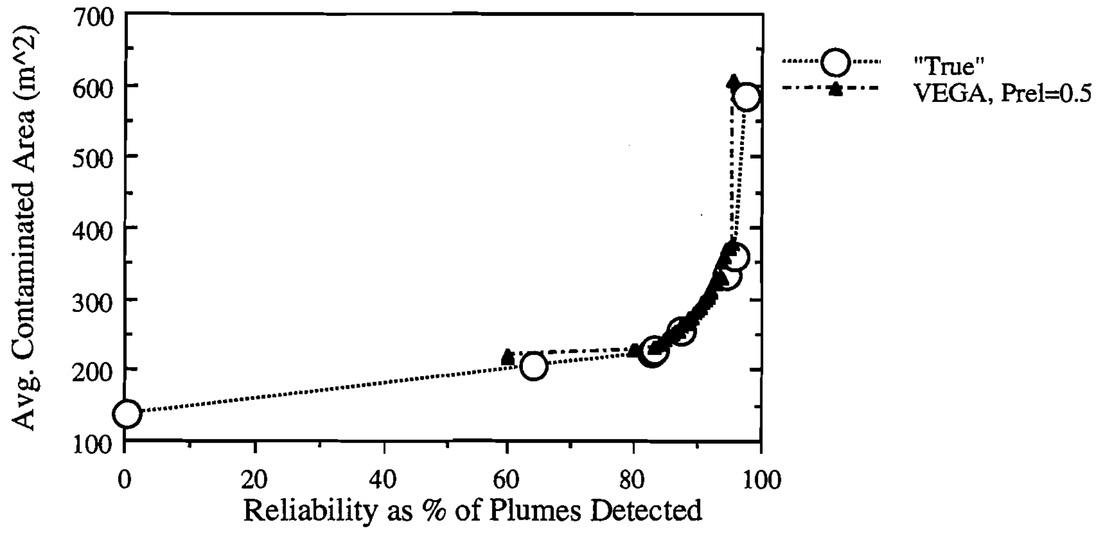

Figure 17. VEGA (Prel=0.5) vs. "True" (Q=10, NPOTW=470)

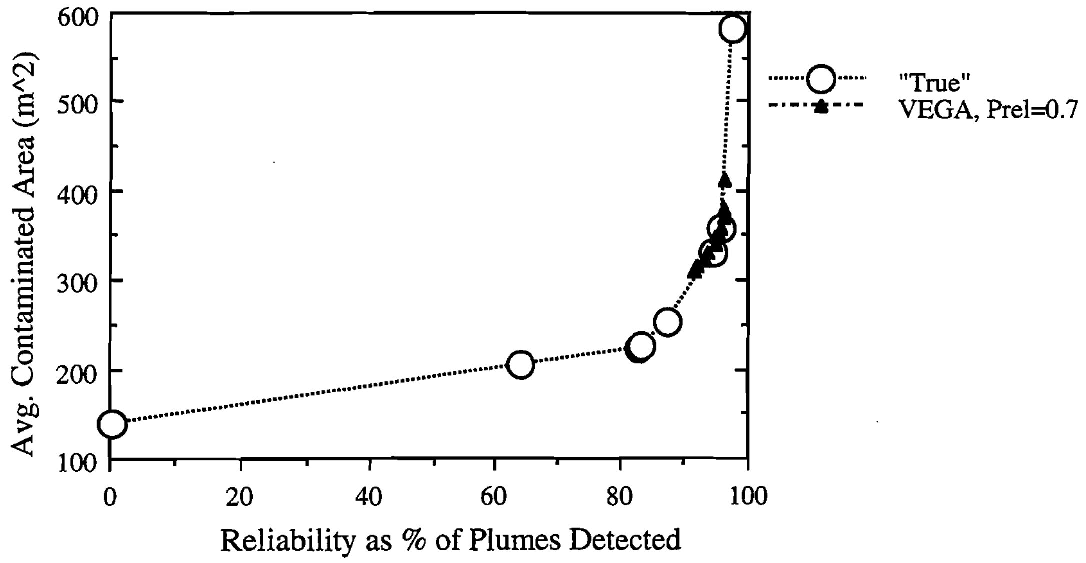

Figure 18. VEGA (Prel=0.7) vs. "True" (Q=10, NPOTW=470)

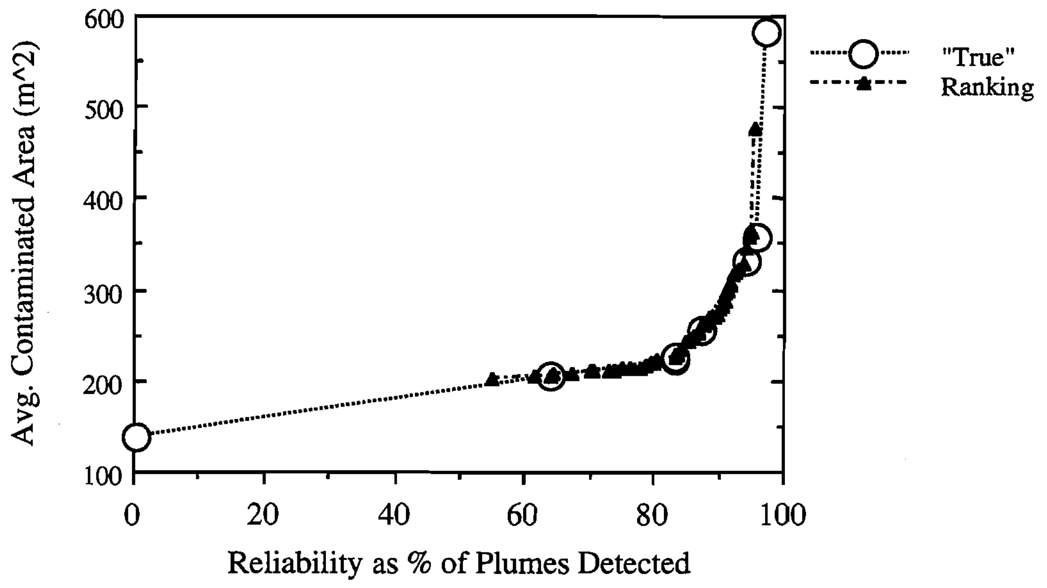

Figure 19. Pareto-Optimal Ranking vs. "True" (Q=10, NPOTW=470)

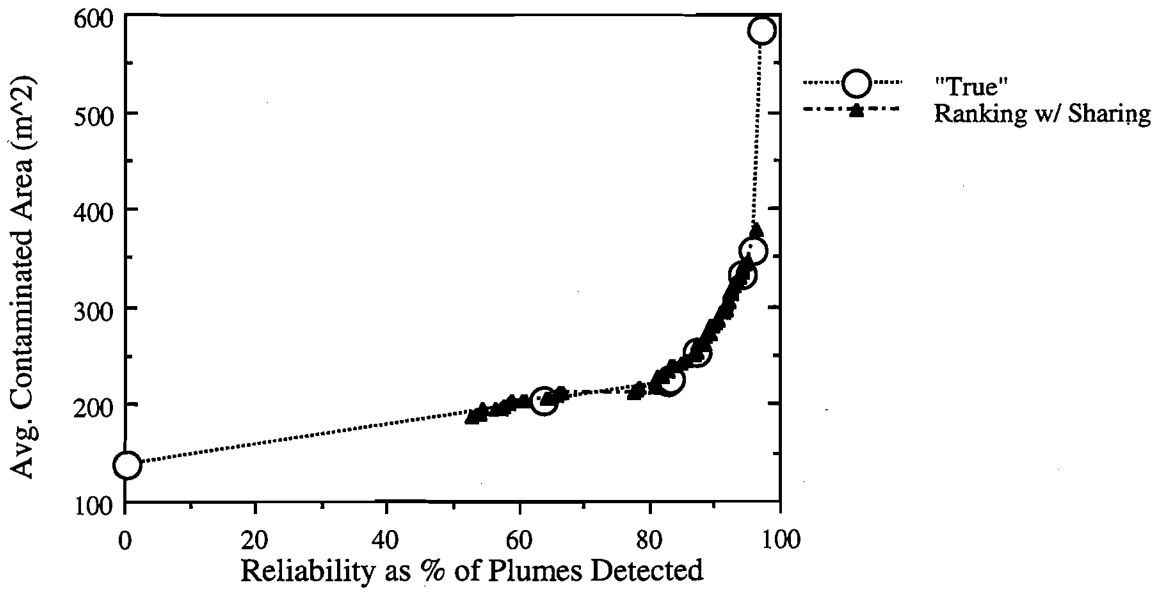

Figure 20. Pareto-Optimal Ranking w/ Sharing vs. "True" (Q=10, NPOTW=470)

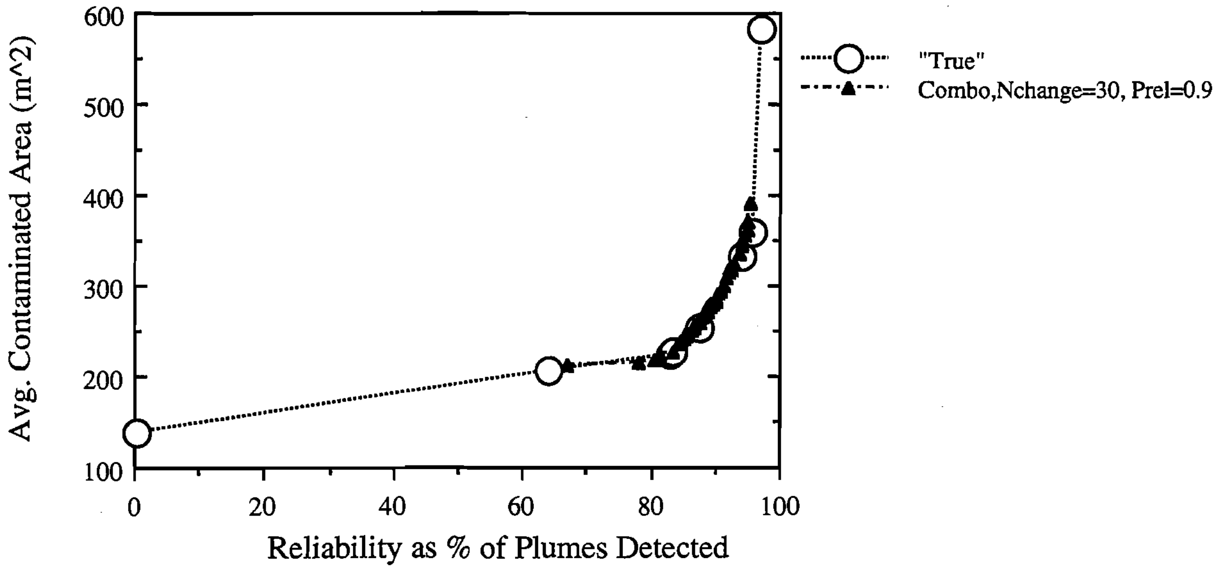

Figure 21. Combination (Nchange=30, Prel=0.9) vs. "True" (Q=10, NPOTW=470)

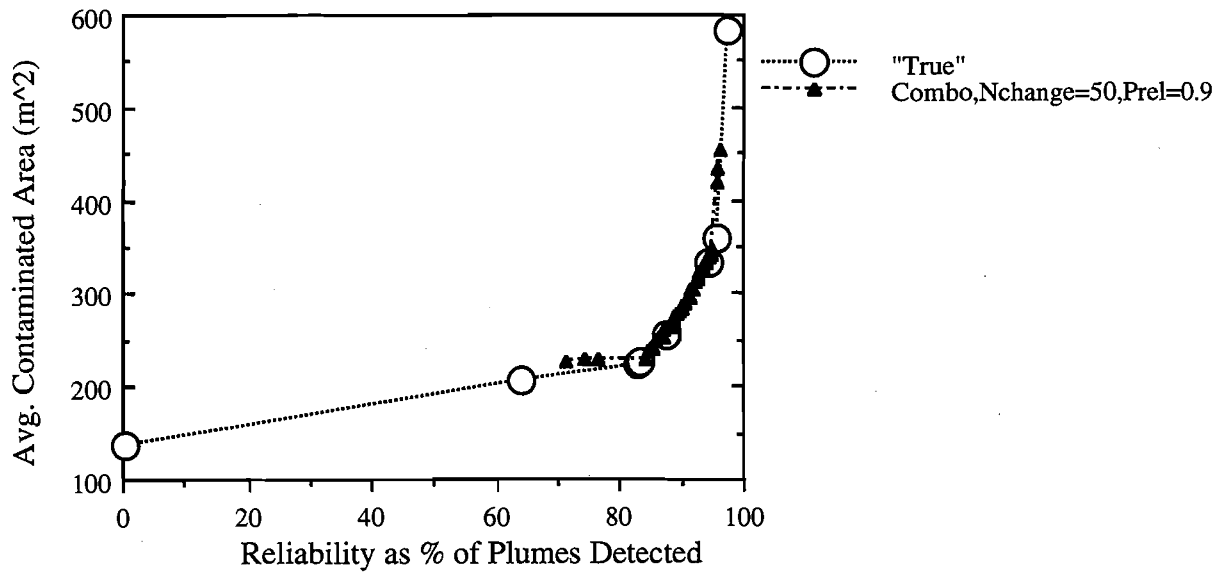

Figure 22. Combination (Nchange=50, Prel=0.9) vs. "True" (Q=10, NPOTW=470)

Chapter 7

Final Remarks

7.1 Research Contributions

This paper has investigated the use of four different formulations of multi-objective GAs for solving a real-world groundwater monitoring problem. It represents one of the first efforts to compare the results of a multi-objective GA to a predetermined tradeoff curve for a complex real-world example problem. The GA-based solution method was shown to be able to generate both convex and non-convex points of the tradeoff curve, accommodate non-linearities in the two objective functions, and find several Pareto-optimal points in a single iteration. The research also introduced the combination formulation which attempts to capitalize on the individual strengths of the VEGA and Pareto-optimal ranking formulations. Finally the effects of sharing in conjunction with Pareto-optimal ranking were investigated to determine the effects sharing has on maintaining the diversity of the GA population and the performance of the GA in generating the multi-objective tradeoff curve.

The use of genetic algorithms to generate the tradeoff curve for the multi-objective groundwater monitoring problem yields both encouraging and challenging results. Although the GAs were able to generate a large number of convex and non-convex points along the tradeoff curve without resorting to a weighted objective function, none of the four multi-objective GA formulations were able to generate the entire tradeoff curve for either of the example data set for *Meyer et al's.* [1992] EPMP. Even after manipulation of the parameter settings, no single GA formulation was able to generate the entire tradeoff curve.

The fact that the GAs were able to generate large portions of the tradeoff curve and suffered only at the ends of the tradeoff curve is encouraging news. Often times decision makers are interested in determining a compromise solution to a given problem. Under these conditions the decision maker will not be interested in the very ends of the tradeoff curve because one objective often must be severely degraded in order to obtain a small improvement in the other.

Economists refer to this type of phenomenon as a case of diminishing returns. If a decision maker is interested in a compromise solution, multi-objective GAs are a very promising tool for generating a tradeoff curve.

All four multi-objective GA formulations were able to generate large portions of the tradeoff curves. Of the four formulations VEGA probably performed worst, although VEGA does allow us to focus at the ends of the tradeoff curve by manipulation of the P_{rel} parameter. The Pareto-optimal ranking formulation generated all but the very highest and very lowest reliability end of the tradeoff curve without requiring the adjustment of any new parameters. The combination formulation was able to take advantage of the strengths of the VEGA and ranking methods to obtain slightly better results at the high reliability end of the tradeoff curve without loss of performance at the low reliability end.

Of the four multi-objective GA formulations the Pareto-optimal Ranking with Sharing Formulation seems most promising for further research. This formulation produces results as good or better than the other three formulations and has one added advantage; diversity in its population even after many generations. This diversity will be of great benefit to the GA for use in interactive decision making, an area that should be specifically targeted for future research.

7.2 Recommendations for Future Research

Just like natural systems need genetic and physical diversity in order to adapt to changing environmental conditions, GAs will require a diverse population of solutions after many generations if they are to be able to adjust to changing objective functions. I believe that *Fonseca and Fleming* [1993] seem to have only scratched the surface of GAs ability to incorporate a decision makers soft information into a deterministic objective function. They allow the GA to proceed through many generations in an attempt to generate a multi-objective tradeoff curve. After the decision maker gets a good idea of the shape of a tradeoff curve he is able to change the problem constraints in order to target portions of the tradeoff curve that are of particular interest. While *Fonseca and Fleming* [1993] use constraints to direct a GA to a certain portion of the tradeoff curve, GAs have the power to adapt directly to changing objective functions. Therefore, changes in objective function can be used in place of constraints (which are difficult for GAs to handle) to direct the GAs search to a particular region of decision space.

Selection of optimal parameter settings is vital to insure the proper performance of GAs. A simple GA requires the selection of only two parameters, p_{mute} and p_{cross}. However, as we begin adding advanced operators to the GAs we need to determine the settings for more and more parameters. Perhaps the best approach to finding these values is to include the parameters as decision variables in the coding of the string. In this way we would allow the GA to find the optimal parameter settings on its own. This seems to be a logical approach to the problem of parameter selection, since GAs are better suited to the task than we are, and such an approach would eliminate the headache of trial-and-error parameter tuning.

Another area that should receive research attention is the extension of multi-objective GAs to more than two objective functions. *Fonseca and Fleming* [1993] investigated a four objective problem, but they do not compare the GA's performance to a predetermined tradeoff curve. Although it is true that it is more difficult to find a "true" tradeoff curve as we add more objectives to a problem, this is exactly the reason we need to compare GA performance to the performance of other techniques. We must determine if GAs can advance beyond a two-objective problem without the degradation of performance associated with conventional weighting techniques.

Finally, as we use GAs to investigate more and more complicated problems we will find the need to study the performance of GAs on parallel computers. For the example problem in this research with 1000 simulated plumes and 470 potential well locations, a single GA run took over two hours on an HP-UX workstation. It would be very difficult to convince a decision maker to spend two hours in front of a computer to work interactively with the GA to find the portion of the tradeoff curve he is interested in. Running GAs on parallel computers for which they are perfectly suited will address this shortcoming.

References

Church , R. L. and C. ReVelle, "The maximal covering location problem", *Papers of the Registered Scientists Association*, Vol. 32, p. 101-118, 1974.

Cohon, J. L. and D. H. Marks, "A Review and Evaluation of Multiobjective Programming Techniques", *Water Resources Research*, Vol. 11, No. 2, p. 208-220, 1978.

Deb, K. and D. E. Goldberg, "An Investigation of Niche and Species Formation in Genetic Function Optimization", *Proceedings of the Third International Conference of Genetic Algorithms*, 1989.

Freeze, R. A., "A Stochastic-Conceptual Analysis of One-Dimensional Groundwater Flow in Nonuniform Homogeneous Media", *Water Resources Research*, Vol. 11, No. 5, p. 725-742, 1975.

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

Goldberg, D. E. and K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", *Foundations of Genetic Algorithms*, 1989.

Goldberg, D. E., K. Deb, and J. Clark, "Genetic Algorithms, Noise, and the Sizing of Populations", *Complex Systems*, Vol. 6, p. 333-362, 1992.

Holand, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

Liepins, G. E., M. R. Hilliard, J. Richardson, and M. Palmer, "Genetic Algorithms Application to Set Covering and Traveling Salesman Problems", *Operations Research and Artificial Intelligence: The Integration of Problem-Solving Strategies*, Kluwer Academic Press, Boston, 1988.

Mantoglou, A. and J. L. Wilson, "The Turning Bands Method for the Simulation of Random Fields Using Line Generation by a Spectral Method", *Water Resources Research*, Vol. 18, No. 5, p. 1379-1394, 1982.

Massman, J. and R. A. Freeze, "Groundwater Contamination from Waste Management Sites: The Interaction between Risk-based Engineering Design and Regulatory Policy", 1, Methodology, *Water Resources Research*, Vol. 23, No. 2, p. 351-367, 1987.

McKinney, D. C. and M. Lin, "Genetic Algorithms in Groundwater Flow Optimization", *EOS, Transactions of the American Geological Union*, Vol. 73, No. 43, 1992.

Meyer, P.D. and E. D. Brill, "A Method for Locating Wells in a Groundwater Monitoring Network Under Conditions of Uncertainty", *Water Resources Research*, Vol. 24, No. 8, p 1277-1282,1988.

Meyer, P. D., The Optimal Design of Groundwater Quality Monitoring Networks Under Conditions of Uncertainty, Doctoral Thesis, University of Illinois at Urbana-Champaign, 1992.

Meyer, P. D., J. W. Eheart, S. Ranjithan, and A. J. Valocchi, "Design of Groundwater Monitoring Networks for Landfills", *ASTSWMO National Solid Waste Forum,* 1990.

Meyer, P. D., S. Ranjithan, J. W. Eheart, and A. J. Valocchi, "Groundwater Monitoring Network Design at Hazardous Waste Disposal Facilities Under Conditions of Uncertainty", *HWRIC Project Report 91-061*, 1992.

Oei, C. K., D. E. Goldberg and S. J. Chang, "Tournament Selection, Niching, and the Preservation of Diversity", *IlliGAL Report 91011*, 1991.

Ranjithan, S., J. W. Eheart, and J. Liebman, "Incorporating Fixed-Cost Component of Pumping into Stochastic Groundwater Management: A Genetic Algorithm-based Optimization Approach", *EOS, Transactions of the American Geological Union*, Vol. 73, No. 14,1992.

ReVelle, C. and R. W. Swain, "Central Facilities Location", *Geographical Analysis*, Vol. 2, p 30-42, 1970.

Schaffer, J. D., "Multiple Objective Optimization with Vector Evaluated Genectic Algorithms", *Proceeding of an International Conference on Genetic Algorithms and Their Applications*, p. 93-100, 1985.

Steuer, R. E., *Multiple Criteria Optimization: Theory, Computation, and Application, John Wiley and Sons, Inc.*, New York, 1986.

Sudicky, E. A., "The Laplace Transform Galerkin Technique: A Time-continuous Finite Element Theory and Application to Mass Transport in Groundwater", Water Resources Research, Vol. 25, No. 8, p. 1833-1846, 1989.

Wang, Q. J., "The Genetic Algorithm and its Application to Calibration Conceptual Rainfall-Runoff Models", *Water Resources Research*, Vol. 27, No. 9, p 2467-2471, September, 1991.

Zimmerman, D. A. and J. L. Wilson, *Description of and User's Manual for TUBA, A Computer Code for Generating Two-Dimensional Random Fields via the Turning Bands Method*, Geoscience Department, New Mexico Institute of Mining and Technology, 1988.