OPTIMIZING GROUNDWATER REMEDIATION DESIGNS USING DYNAMIC META-MODELS AND GENETIC ALGORITHMS

BY

SHENGQUAN YAN

B. Engr., Tongji University, 1996
M. Engr., Tongji University, 1999

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Environmental Engineering in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

ABSTRACT

Large-scale water resources optimization often involves using time-intensive simulation models to evaluate potential water resource designs or calibrate parameter values. The extensive computational requirements have become a critical issue for applying global optimization algorithms to water resources problems. In order to improve the computational efficiency, this research developed a dynamic modeling framework called Adaptive Meta-model Genetic Algorithm (AMGA), in which time-efficient meta-models are automatically trained and adaptively retrained within a genetic algorithm (GA) to replace the time-intensive simulation models. A dynamic learning approach is proposed to periodically sample new solutions both to update the meta-models and to correct the GA's convergence. Different configurations of AMGA were tested on a hypothetical groundwater remediation design case, and then the best configuration was applied to a field-scale remediation site. In these applications, AMGA saved 85- 90% percent of the simulation model calls with little loss in accuracy of the optimal solutions. These results show that the method has substantial promise for reducing computational effort associated with large-scale water resources optimization.

The prediction accuracy of meta-models is a key factor in determining AMGA's performance. This research then developed an advanced meta-model construction method to achieve improved predictions for both water resources optimization and forecasting problems. In this research we used a trust region-based meta-model approach, in which a global model and a set of local models are constructed into a hierarchical ensemble. A trust region testing strategy selects the most appropriate sub-models, and these model predictions are blended by a gating network to generate the final prediction. The

technique was tested on the field-scale remediation case study. Our results show that the adaptive GA coupled with the trust region-based meta-models converged with somewhat higher accuracy and identified the optimal solutions faster. Finally the technique was tested on a nitrate concentration prediction problem. The results show that the trust region-based prediction models had better prediction performance than the corresponding single-prediction models.

Real-world optimization problems are often inherently uncertain. The last focus of the research is to extend the adaptive modeling techniques in a stochastic optimization framework so that robust optimal solutions can be efficiently identified in the presence of parameter uncertainty. The developed algorithm, called Noisy-AMGA, minimizes the expected fitness function with a constrained reliability level. The algorithm incorporates an archiving cache and an incremental Latin Hypercube sampling technique so that the estimation of the fitness expectations can be progressively improved. As in AMGA, the meta-models in Noisy-AMGA are online updated but they are trained to predict the expected outputs. The method was applied to both the hypothetical remediation case study and the field-scale case study, where the primary source of uncertainty stems from hydraulic conductivity values in the aquifers. Our results show that the technique can lead to far more reliable solutions with significantly less computational effort.

DEDICATED TO MY WIFE MENGYUAN, AND ALL MY FAMILY.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Barbara Minsker, for her outstanding guidance and support throughout my graduate studies in UIUC. She is a brilliant mentor and she inspired me numerous times in the long journey of pursuing mental improvement. I especially appreciate the intellectual freedom she endowed me with, encouraging me to think deeply and critically about my research and allowing me to develop interdisciplinary research interests.

I would also like to thank the members of my thesis committee, Professor Wayland Eheart, Professor Albert Valocchi and Professor Dan Roth, for their valuable technical assistance and helpful discussions.

I wish to thank Professor Sylvian Ray for assisting me in setting up the first neural network. I also wish to thank Professor Barbara Bailey for answering me many statistical questions.

My thanks go out to Professor Wayland Eheart and Professor Ximing Cai for providing me advice and comments during the last phase of my research.

I would like to thank Dr. Jian-ping Suen for generously providing me the Sangamon case data set and his model results.

I would like to acknowledge the U. S. Army Research Office under grant number DAAD19-001-1-0025 for funding this research. I would also like to acknowledge the U.S Geological Survey for sharing the Samgamon case data set to researchers.

I would also like to thank Professor Vernon Snoeyink and the CASE members in helping me improve my presentation skills.

I am indebted to Dr. Bruce Loftis, for his help in initiating the distributed computing infrastructure. I also would like to acknowledge Tom Roney, who generously allowed me to use TRECC and CCG clusters at NCSA.

A special thank you to my fellow group-members at the EMSA research group. Dr. Felipe P. Espinoza, you helped me setup two test cases in my research. Meghna Barbbar, you helped me with my job search and brought me many joyful moments. Xiaolin Ren, you are a great friend at all times. Abhishek Signh, I thank you in particular for your countless help in my research, and the inspiring conversions between you and me are unforgettable.

This dissertation would not have been possible without my dearest wife, Mengyuan. Her love and support accompanied me through the difficult times. She is the most precious gift I have been given in this world.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Many human activities, such as agriculture practices, landfills, and military operations, have directly or indirectly caused the contamination of groundwater. Remediating these contaminated sites has been a costly but urgent problem in the United States as the contaminated groundwater is a potential public health hazard. The U.S Environmental Protection Agency (U.S. EPA, 1997) has reported that the Department of Defense estimates that more than 8,000 sites remain to be remediated on over 1500 installations as of 1995. These sites are Army, Air Force, Navy Bases and formerly-used defense sites. The total cost associated with remediation actions for these sites is estimated to be up to $29 billion in 1996 dollars. In an earlier report, the National Research Council (NRC) estimates that there are approximately 400,000 contaminated sites in the United States, and the total cost for cleaning up these sites is up to $1 trillion (National Research Council, 1994).

Pump-and-treat systems, coupled with other remediation technologies for source control, have been traditionally used to clean up contaminated sites (U.S. EPA, 1997). A number of enhanced pump-and-treat options are also being used or investigated, such as in situ bioremediation, monitored natural attenuation, and air sparging. Although the pump-and-treat option has been widely accepted as a de-facto standard in remediating contaminated sites, the pump-and-treat option is often criticized for the high cost associated with the technology. The major cost components of a pump-and-treat system

consist of the pumping and monitoring well installation cost and the long term operation and maintenance cost of the installed wells. Given the scope of the contamination and the vast amount of money involved in a remediation process, improved design solutions for pump-and-treat systems are needed.

The advance of modeling techniques and computing facilities has made it possible to simulate subsurface flows and chemical transport processes in contaminated aquifers. Simulation models are usually complex solutions to partial differential equations describing the underlying physical advection and diffusion processes. With the help of the simulation models, finding improved design solutions for pump-and-treat systems can be done by the trial-and-error method. This method generally requires expert knowledge to set up a number of potential trial designs, and the simulation models can be applied thereafter to assess the quality of the trial designs so that the least cost alternative can be selected. The trial-and-error approach, unfortunately, is very inefficient and unlikely to identify the best design(s). This is because the decision spaces of remediation design problems usually consist of a large number of continuous and discrete decision variables, such as pumping rates, pumping well locations, and flags of well operations. As a result, finding the optimal solution in a huge decision space by trial-and-error approaches is unrealistic.

Many previous studies have focused on coupling optimization tools with simulation models to search for effective remediation designs. Nonlinear programming and heuristic optimization are the commonly used optimization methods. Nonlinear

programming requires convex objective functions and derivative information for the objective function and the constraints, which is difficult to obtain if complex simulation models and discrete decision variables are involved. As a result, the Genetic Algorithm, a heuristic optimization method that simulates the biological evolution process by selection, crossover and mutation operators, has become a popular method of choice in recent years. Compared to the traditional nonlinear programming methods, GAs require neither continuity nor convexity of the objective function and the constraints, nor-indeed, that they be closed-from functions of the decision variables. GAs have been shown to be effective and robust for identifying optimal design solutions in the discrete decision spaces of remediation problems. However, the main disadvantage of GAs is the computational cost. GAs require much more objective function evaluations during the slow evolving process, particularly when the decision space is large and discrete. This limitation is a major barrier when applying GAs to field-scale remediation design problems because the objective function evaluations often require time-intensive solutions of complex numerical models.

1.1 Objectives and Scope

The objective of this research is to reduce the computational overhead when solving remediation design problems, and other water resources optimization problems that require time-intensive simulation models, using GAs. A key to accelerating GAs' computation is to replace most of the numerical model evaluations by computationally

cheap surrogate (meta-model) evaluations. In the groundwater management domain, the meta-models are usually regression models or machine learning models that are trained to learn the response mechanism of the numerical models. Once they are trained, they provide faster predictions of the flow and transport processes. In previous research, this method has been used to speed up GAs' computation using a static training and prediction approach. Using this approach the meta-models are trained before the GA starts, and the meta-models are used to replace the numerical models without retraining in the optimization process. This approach, however, may compromise GAs' global searching ability because the statically trained meta-models may not have good prediction accuracy at local regions and may cause the GAs to converge to local optima that are not globally optimal. In order to overcome this problem, this research explores a dynamic approach that adaptively trains and retrains meta-models embedded into a GA optimization framework. As the meta-models are retrained, their prediction accuracy within local regions is improved. Therefore this framework is helpful for avoiding convergence to local optima and attaining the globally optimal or near-optimal solutions. The specific goals of this research are to:

- Design a dynamic meta-model GA framework that can be used to solve optimal groundwater remediation design problems, as well as other water resources optimization problems with time-intensive simulations, efficiently and effectively. The framework will be tested on both a hypothetical remediation case study and a field-scale remediation case study.

- Design advanced prediction meta-models that can provide more accurate predictions. The method will be tested with the dynamic GA framework on the field-scale remediation case study, as well as on a general nitrate concentration prediction problem.
- Modify the dynamic optimization framework to handle uncertainty. Uncertainty stemming from uncertain parameters (in this case, hydraulic conductivities) will be taken into account in the optimization process so that reliable optimal solutions can be obtained using the dynamic optimization framework.

1.2 Summary of Research Approach

The main goal of the research is to reduce the computational cost associated with GAs in groundwater remediation optimization and other time-intensive water resources applications. To accomplish this goal, this research developed a new framework (Chapter 4) in which dynamically updated meta-models, primarily machine learning models, are embedded into a Genetic Algorithm to replace time-intensive simulation models. In the framework, a sampling technique is the key that threads the meta-models and the GA together so that they collaborate in identifying optimal solutions. The dynamic GA approach is tested with two remediation case studies presented in Chapter 3, which are used thoroughly as test beds for this research.

In Chapter 5, an advanced meta-model approach based on the concept of divide-and-conquer is presented. This concept attempts to tackle a complex problem by dividing

it into smaller and simpler sub-problems and solving them individually. A similar concept is used to construct a global prediction model and a series of local prediction models hierarchically. These trust region are used to select appropriate sub-models when new inputs arrive, and the outputs of the trusted sub-models are then blended by a gating neural network that generates the coefficients to mix the sub-model predictions. The performance of the hierarchical meta-models is tested using the dynamic GA framework developed in Chapter 4. Then the hierarchical modeling approach is tested on a stream nitrate prediction problem.

Chapter 6 further develops the dynamic GA framework so that it can be applied in a noisy environment. The chapter assumes that uncertainty is present due to incomplete knowledge of the characteristics of parameters such as hydraulic conductivity values. A noisy dynamic GA that minimizes the expected cost is proposed. In the noisy dynamic GA, the meta-models are used to predict the expectations of the corresponding fitness components and the reliability levels of the trial designs, and statistical tests are used to select appropriate individuals in a tournament selection scheme. The noisy dynamic GA is tested using the two remediation case studies presented in Chapter 3, and the results are presented in Chapter 6. In the following sections, a more detailed summary of each chapter is provided.

1.2.1 Chapter 2: Literature Review

Chapter 2 provides a summary of previous approaches that have been used to tackle groundwater remediation design problems. In the water resources field, many

researchers have used gradient based optimization algorithms, such as linear programming, non-linear programming, dynamic programming, and variants of these approaches, to search for optimal remediation designs. The gradient based optimization algorithms are generally local searching algorithms, which are fast but easily trapped into local minima. The global optimization algorithms, such as simulated annealing, genetic algorithms, tabu search, and ant colony optimization, have become popular in the last two decades due to the advance of information technology and the speedup of computing facilities. The global optimization algorithms are heuristic searching algorithms that have fewer requirements for problem structure but are more computationally intensive. Recently, a couple of approaches have been proposed to develop efficient global optimization algorithms, such as meta-model techniques that use fast approximate models to replace slow simulation models. The chapter reviews the efficient global optimization approaches with a focus on the applications of the meta-model approach in water resources and optimization fields. Then, the chapter covers advanced meta-model methods such as hierarchical mixture models and trust region based meta-model management. The chapter also reviews stochastic optimization methods that have been used to optimize groundwater remediation designs under uncertainty. During the discussion, the chapter raises several issues that motivate this research, which will be addressed in detail in subsequent chapters.

1.2.2 Chapter 3: Groundwater Remediation Example Cases

Chapter 3 presents two groundwater remediation example cases in detail. The two groundwater remediation cases are the test beds for methodologies addressed in chapters 4, 5 and 6. The first groundwater remediation case is a hypothetical case designed by Smalley et al. (2000). The design objective of the hypothetical case is to minimize the costs associated with pumping well installations and operations so that the human exposure risk due to contaminant migration can be controlled within a specified level. The hypothetical case incorporates flow and transport simulation models but the evaluation of the simulation models is relatively fast. This enables extensive testing of the methodologies proposed in later chapters before they are applied to more computationally-intensive cases. The second case is a field-scale remediation case study at Umatilla, Oregon. The case was developed by the Army Corps of Engineers (ACE) to remediate chemical contaminants resulting from onsite explosives washout operations and groundwater migrations. Three formulations have been developed for this site and the first two are discussed in detail in this chapter. The flow and transport simulations of the Umatilla case are based on a finely divided grid and hence the evaluation of the simulation models is much more expensive than for the hypothetical case. The Umatilla case, nevertheless, is a more realistic case and the performance of the proposed methodologies reflects their performance in real world applications.

1.2.3 Chapter 4: An Adaptive Meta-model Genetic Algorithm

Although approximate models have been attempted for improving computational efficiency of GAs for groundwater remediation problems, in most instances, multiple simulation runs have been done prior to the optimization. Chapter 3 demonstrates that the statically trained meta-models can lead to sub-optimal solutions in the dynamic environment of a GA. This chapter develops a dynamic modeling framework called Adaptive Meta-model Genetic Algorithm (AMGA), in which artificial neural networks are used as meta-models and they are automatically trained and adaptively retrained within a genetic algorithm (GA) to replace the time-intensive flow and transport simulation models. A dynamic learning approach is proposed to periodically sample new solutions both to update the ANNs and to correct the GA's convergence. In addition, AMGA incorporates an efficient caching technique to store and retrieve a history of simulation results, and AMGA is linked with a distributed computing infrastructure so that the simulation model evaluations are executed in parallel. Different configurations of AMGA were tested on the hypothetical groundwater remediation design case, and then the best configuration was applied to the Umatilla remediation site. In these applications, AMGA saved 85- 90% percent of the simulation model calls with little loss in accuracy of the optimal solutions. These results show that the method has substantial promise for reducing computational effort associated with large-scale water resources optimization.

1.2.4 Chapter 5: Trust Region-based Meta-models for Water Resources

Chapter 5 presents a more advanced meta-model construction approach that can achieve improved predictions in the adaptive GA framework developed in Chapter 4. In this chapter, we propose a trust region-based meta-model modeling approach, in which a global model and a set of local models are constructed into a hierarchical ensemble to replace the time-consuming simulation models in the GA framework. The hierarchical meta-model ensemble is also dynamically trained and adaptively updated. During each updating period, a multi-sampling technique is used to cluster the collected numerical solutions into hierarchical training regions and establish the global and local sub-models from these clustered regions. A trust region testing strategy selects the most appropriate sub-models, and these model predictions are fed to a gating network that yields the final prediction by mixing the trusted model outputs using appropriate mixing (weight) coefficients. This allows the local regions (particularly those near the optimal solution) to be approximated by smoother and smaller meta-models with higher accuracy, which in turn can speed up GA's convergence when the population moves into local regions. The technique was tested with artificial neural networks (ANNs) and support vector machines (SVMs) on the Umatilla case study. The results show that the adaptive GA coupled with the trust-region-based meta-models converges with somewhat higher accuracy and identifies the optimal solutions faster. Finally the technique was tested on a nitrate concentration prediction problem. The results show that the trust region-based prediction

models have better prediction performance than the corresponding single-prediction models.

1.2.5 Chapter 6: Optimizing Groundwater Remediation Designs Under Uncertainty

The presence of uncertainty in real-world optimization applications is ubiquitous. The computational issue is compounded when optimizing under uncertainty, since Monte Carlo simulations are often required to evaluate objective function values over multiple parameter realizations. In order to improve the computational efficiency, Chapter 6 presents a new algorithm that applies AMGA in a noisy environment. The algorithm, called Noisy-AMGA, uses artificial neural-networks as meta-models. However, the meta-models are trained online to predict the expectations of the objectives, using sampling results created during the GA run. As in AMGA, the meta-models are adaptively updated to improve their prediction performance and correct the GA's convergence as the search progresses. Latin Hypercube sampling of hydraulic conductivities is used to efficiently sample parameters for the Monte Carlo simulation and the sampling results are archived so that the estimation of the objective function expectations is progressively improved. The GA is modified to incorporate hypothesis tests to produce reliable solutions. The method is applied to both the hypothetical remediation case study and the Umatilla case study, where the primary source of uncertainty stems from hydraulic conductivity values in the aquifers. Our results show that the technique can lead to far more reliable solutions with significantly less computational effort.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Chapter 1 presented the background of groundwater remediation and the research objectives. In this chapter, a literature review is provided. The review covers previous research in optimal groundwater remediation design and approaches for applying them in the presence of uncertainty.

2.2 Optimization Methods for Groundwater Remediation Designs

2.2.1 Gradient-based Local Optimization Methods

Application of pump-and-treat systems requires optimization techniques that can find cost-effective design solutions to groundwater remediation sites. Traditional gradient based optimization methods have been used extensively in a number of studies. The gradient based optimization methods are generally local optimization algorithms. They require the evaluation of the first order (the Jacobson matrix) and/or the second order (the Hessian matrix) derivatives so that a local search direction can be obtained following the gradient direction (for non-linear programming), or the local search dimensionality can be reduced (for differential dynamic programming). The local search terminates at local minima because no better search directions can be found once local minima are reached. Therefore the local optimization algorithms rely on convexity of both the objective functions and the constraints in order to achieve global minimum. Application examples

using the optimization approaches include: nonlinear programming (Gorelick et al., 1984; Ahfeld et al., 1988; Bear and Sun, 1998; and McKinney and Lin, 1996), quadratic programming (Lefkoff and Gorelick, 1986), differential dynamic programming and variants (Chang et al., 1992; Culver and Shoemaker, 1992, 1993, 1997; Whiffen, 1995; Mansfield et al., 1998; and Minsker and Shoemaker, 1998a, 1998b), outer approximation (Karatzas and Pinder, 1993), mixed-integer nonlinear programming (McKinney and Lin, 1995), cutting plane (Karatzas and Pinder, 1996), and multiscale combined with differential dynamic programming (Liu, 2001; Liu et al., 2001; and Liu and Minsker, 2002).

2.2.2 Global Optimization Methods

Although the local optimization algorithms are efficient at finding optimal solutions under convex assumptions, their application to real-world water resource problems, such as optimal groundwater remediation designs, is not always the best choice. This is because solving water resource optimization problems often requires coupling a mixed-integer optimization algorithm with complex simulation models to evaluate potential solutions (e.g., groundwater flow and transport models). These coupled models and discrete decision variables can create an optimization problem that is non-convex, discontinuous, or discrete, and can be difficult to solve with local optimization algorithms. As a result, alternative global optimization algorithms have been investigated in recent years because of their fewer requirements on gradient information and their better global searching ability.

The global optimization algorithms usually utilize heuristic searching techniques to find improved solutions. The typical global optimization algorithms that have been used in water resources field are simulated annealing and genetic algorithms. For example, simulated annealing (Dougherty and Marryott, 1991; Rizzo and Dougherty, 1996); genetic algorithm (Eheart et al, 1993; Ritzel et al, 1994; McKinney and Lin, 1994; Aly and Peralta, 1999; Yong and Shoemaker, 1999; Smalley et al, 2000); ant colony (Chan Hilton and Li, 2005); and tabu search (Zheng, 1996). GAs incorporating noisy and multiple objectives are also proposed (Reed et al., 2001, 2003; Gopalakrishnan et al., 2003, Singh and Minsker, 2001, 2003).

2.3 Meta-Model Modeling in Optimization

While the global optimization algorithms are more robust at attacking complex, discontinuous remediation design problems, they are, unfortunately, often more inefficient: when no gradient information is available, the global optimization methods rely on far more design evaluations (using simulation models) to explore the topology of the search space and to slowly converge toward the global minimum. For remediation design problems that are often associated with time-intensive flow and transport models, this makes the optimization time requirement unrealistic for large field-scale cases.

One remedy for alleviating the computational burden is to create efficient approximation meta-models to replace the simulation models. Approximation meta-models usually involve using learning techniques to learn the response surface of a time-

intensive model from a training set. Then the meta-models can replace the expensive simulation models to speed up the optimization. A number of different approximation models have been applied in water resources management in recent years. Cooper et al., 1998, for example, used curve-fitting methods to approximate the response surface. Vermeulen et al., 2002, used empirical orthogonal functions to reduce model dimensionalities. A popular approach, however, is to use machine learning models, such as artificial neural networks (ANNs) or support vector machines (SVMs), as approximate meta-models. The details of the two models are described below.

2.3.1 Meta-models

Artificial Neural Networks (ANNs). Artificial Neural Networks (ANNs) have been successfully applied to numerous water resource applications, including Garrett et al. (1992), Ranjithan et al. (1993), Rizzo and Dougherty (1996), Rogers and Dowla (1994), Rogers et al. (1995), Aly and Peralta (1999), Maskey et al. (2000), Dibike and Solomatine (2001), Coppola et al. (2003) and others. Their widespread application in different disciplines can be attributed to their ability to approximate almost any linear or non-linear function. As shown in Figure 2.1, an ANN has many neuron-like units. Each unit accepts inputs and gives an output according to its activation function. The interconnections among the neurons in an ANN have represented weight values, which compose a large parameter set. By gradually tuning the weights associated with each interconnection, a supervised learning algorithm can learn the mapping relationship

between the inputs and the outputs sampled from a training set. The trained ANN is then used to make a prediction (Russel and Norvig, 1995).

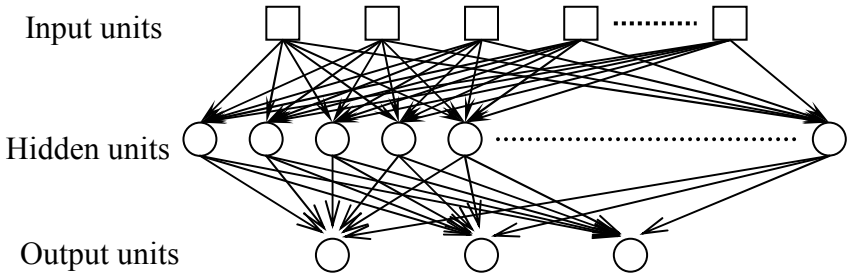

Figure 2.1 ANN architecture.

Support Vector Machine (SVM). In many respects, SVMs showed competing or even better performance than ANNs (Schölkopf, 1999). Applications of SVMs in the water resources field include Liong and Sivapragasam, 2002, Khan and Coulibaly, 2006, and Asefal et al, 2005a and 2005b. The basic principle of SVM is demonstrated by a simple linear separable classification problem (Figure 2.2), where the white nodes and black nodes represent the positive (+1) and negative points (-1) in a hyperspace. The objective is to find a hyperplane $\mathbf{w} \bullet \mathbf{x} + b = 0$ separating the two classes of points. The SVM algorithm looks for the separating hyperplane with the largest margin (the perpendicular distance between the two dashed lines), which is the distance that the hyperplane shifts in a perpendicular direction without losing its separation ability.

Intuitively, the minimum closest points (the circled points) defining the separating hyperplane are "support vectors", since they are confining the margin and moving any of them will change the hyperplane's normal vector **w**. Note that although two support

vectors are enough for defining a hyperplane in a two dimensional space, there are usually more than two support vectors in a high dimension hyperspace.

The SVM training algorithm uses a Lagrangian formulation to maximize the margin. While this illustrative example is for classification of data, SVMs can also be used to fit a nonlinear function to data as in this research. For nonlinear function approximation, SVM used in this work finds a kernel function that maps the inseparable input data to a higher dimensional hyperspace. Through this mapping, the data become more separable, allowing separating hyperplanes to be found to classify the data into similar groups (as illustrated in Figure 2.3). Support vectors are then found for each class, as described previously. The support vectors are used to approximate the function represented by the data.

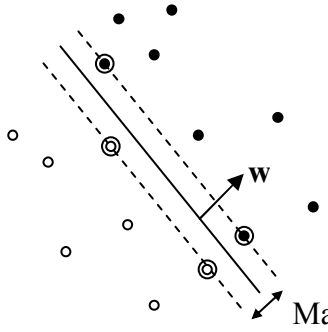

Figure 2.2. Maximizing the margin of separating hyperplanes; the circled points are support vectors.

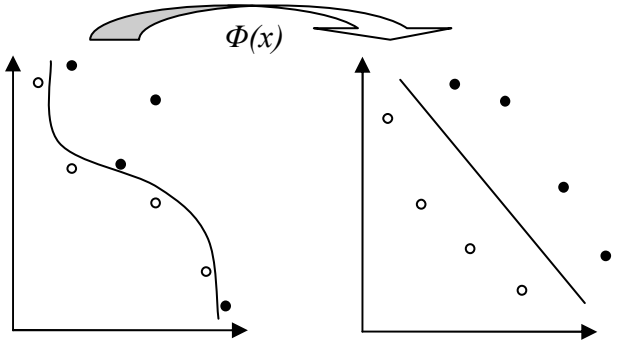

Figure 2.3. Mapping from a inseparable space to a separable space

2.3.2 Meta-model Modeling Techniques

The approximation models used in the above applications all use offline training to statically train the meta-models. The simulation models are run offline (i.e., prior to the optimization run) a number of times to generate the training set. Then the meta-models are trained only once and are never updated subsequently. The disadvantage of this approach is that the statically trained meta-models are usually trained on a globally sampled training set. For a complex search space with multiple local minima, which exist in many remediation design problems, a meta-model may have poor accuracy when the optimization algorithm searches solutions within local regions. The discrepancy can mislead the optimization into false optima due to the increased error of the meta-models.

To alleviate this difficulty, a few studies have proposed adaptive response surface approaches in recent years. Brooker (1998) proposed a framework for generating and managing a sequence of meta-models to the objective function. Jin et al. (2002) used Evolutionary Algorithms (EAs) with ANNs that adapt to sampling model errors to accelerate aerodynamic design problems. However, none of these studies have focused on meta-model models within GAs and there are no published applications of these methods in the water resource field. One of the researchers in the ESTCP project mentioned previously did use an online training approach (Peralta, 2002), but details were not available at the time of this publication to determine whether the approach was adaptive.

Another promising approach to handle the issue is the divide-and-conquer method. The divide-and-conquer method attacks a complex problem by dividing it into smaller and easier-to-solve subspaces and then combining the sub-solutions to yield the solution to the original problem. The decision-tree is an example of such an algorithm that continually classifies training points into sub-spaces until reasonable prediction can be made. The idea can actually be generalized for function approximations. Recently, Jacob and Jordan (1993) and Jordan and Jacob (1994) proposed the mixture-of-experts model, which hierarchically blends the outputs of local experts trained by fitting simpler surfaces to sub-divided local data sets. Zhang and Govindaraju (2000) applied modular ANNs using this "mixed experts" method and demonstrated improved predictions over a single ANN for rainfall predictions However, in his application, the local experts are not organized in a hierarchical relationship.

2.3.3 Meta-model Management Using Trust Region Methods

If multiple local models are used as an approximation meta-model, these local approximation models must be properly managed so that they can collaborate in making predictions. In the optimization field, the advance of trust region optimization incited a new approach to managing local models (Alexandrov, 1998). In the trust region optimization framework proposed by Alexandrov, a series of approximation models with different fidelities and trust regions are created. Whenever a prediction is required, the most appropriate model is chosen to predict the reduction in the objective function. The new searching direction and step size are then formed so that the path of the searching

point is confined within the trust region of the chosen model. After the prediction, the approximation model's trust region is adaptively adjusted according to the trust region updating rules. The advantage of this framework is that approximation models established for different sub-problems can be coordinated to produce predictions instead of exploiting a single approximation model. This concept can therefore be extended for managing multiple approximation models, in conjunction with the divide-and-conquer method for model creation.

2.4 Optimizing Groundwater Remediation Designs Under Uncertainty

Groundwater remediation designs are often inherently uncertain, due to the inherent natural heterogeneity and our lack of complete knowledge of the involved geophysical and chemical processes. Neglecting the possible uncertainty during optimization could lead to system failure. Therefore it is essential to assess and quantify the degree of uncertainty during the design process so that robust solutions that can tolerate parameter variations may be identified. Uncertainty in groundwater remediation problems primarily stems from uncertain hydraulic parameters and initial concentration distributions. Two approaches have been adopted by researchers to solve such problems. The first approach uses chance-constrained optimization techniques to convert stochastic models into deterministic equivalents. The converted deterministic models are easier to solve because they have less memory and computational requirements than the stochastic models. Applications of the chance-constrained optimization approach include Charnes

and Cooper 1959, Wagner and Gorelick 1987, Gailey and Gorelick 1993, Tiedman and Gorelick 1993, Chan 1994 and Sawyer and Lin 1998.

The second alternative for handling uncertainty is to generate multiple equal likely parameter realizations, using methods such as geostatistical analysis. The objective function and constraints are then evaluated so that the constraints satisfy a given number of realizations that are randomly drawn from these generated realizations. The advantage of the approach is that reliabilities can be easily computed using Monte Carlo simulation techniques instead of resorting to the complex functional relationship between the objective functions and the parameter uncertainty. Application examples of this approach include Wagner and Gorelick 1989, Chan 1993, and Morgan et al 1993.

One such parameter realization approach used in recent years is Noisy Genetic Algorithm (Noisy-GA). Noisy-GAs operate like traditional GAs except that the algorithms use sampling techniques to estimate the objective function expectation caused by uncertain parameters. Noisy-GAs have the same advantages as traditional GAs in handling non-convex, discrete problems that are typical in groundwater remediation management. However the sampling technique makes it less likely to converge to over-optimistic solutions that can easily fail the constraints by small perturbations of the parameters. Applications of Noisy-GAs in the water resources area include Hughes (2001) and Teich (2001), who used sampling to estimate uncertainty distribution; Smalley et al. (2000), who used noisy GAs to find robust solutions for a hypothetical remediation case study; Chan Hilton and Culver (2005) used sampling and aging to find

robust solutions with a specified level of uncertainty; Singh and Minsker (2004) used probabilistic selection and an extended archiving technique to improve the performance of Noisy-GAs for groundwater remediation management.

The Noisy-GA method, like most stochastic optimization methods, suffers from severe computational overhead because it evaluates the objective function by simulating multiple parameter realizations. The meta-model modeling technique used for deterministic optimization is therefore a natural solution for expediting the Noisy-GA's computation. So far, few studies have been done to explore the meta-model modeling techniques in a stochastic optimization method. In 2006, Bau and Mayer used kriging models as meta-model models to optimize a hypothetical groundwater remediation design problem, but they used enumeration to identify optimal solutions. Hence, a general stochastic optimization framework that can incorporate meta-model models and demonstrate its effectiveness for complex field scale remediation design problems is required.

CHAPTER 3

GROUNDWATER REMEDIATION EXAMPLE CASES

3.1 Introduction

The methodologies developed in this work are tested using two groundwater remediation design case studies. The first case is a hypothetical case created by Smalley et al. (2000) and the second case study is a field-scale remediation case study at Umatilla Army Depot, Oregon. Summaries of the two case studies are given in this chapter.

3.2 A Hypothetical Groundwater Remediation Case

The first case study is a groundwater remediation case study used in several previous studies (Smalley et al., 2000; Gopalakrishnan et al., 2003 and Ren & Minsker, 2005). This case study was developed using data derived from the Borden site, which was originally developed by McKay et al. (1986), Freyberg (1986) and Sudicky (1986). As shown in Figure 3.1, the fate and transport of BTEX with a peak initial concentration of 200 mg/L, is modeled using a coarse grid of 16 by 8 elements in an aquifer site of 480 meters by 240 meters in dimensions. The aquifer is heterogeneous, isotropic and confined. Twelve monitoring wells are located at the boundaries of the site and several locations in the source area (see Figure 3.1) for measuring contaminant concentrations and estimating human health risk during the remediation period. Using pump and treat technology, up to three remediation wells with a maximum capacity of 250 m3/day per well are considered for installation, with each well having 58 candidate predetermined potential locations as

shown in Figure 3.1.

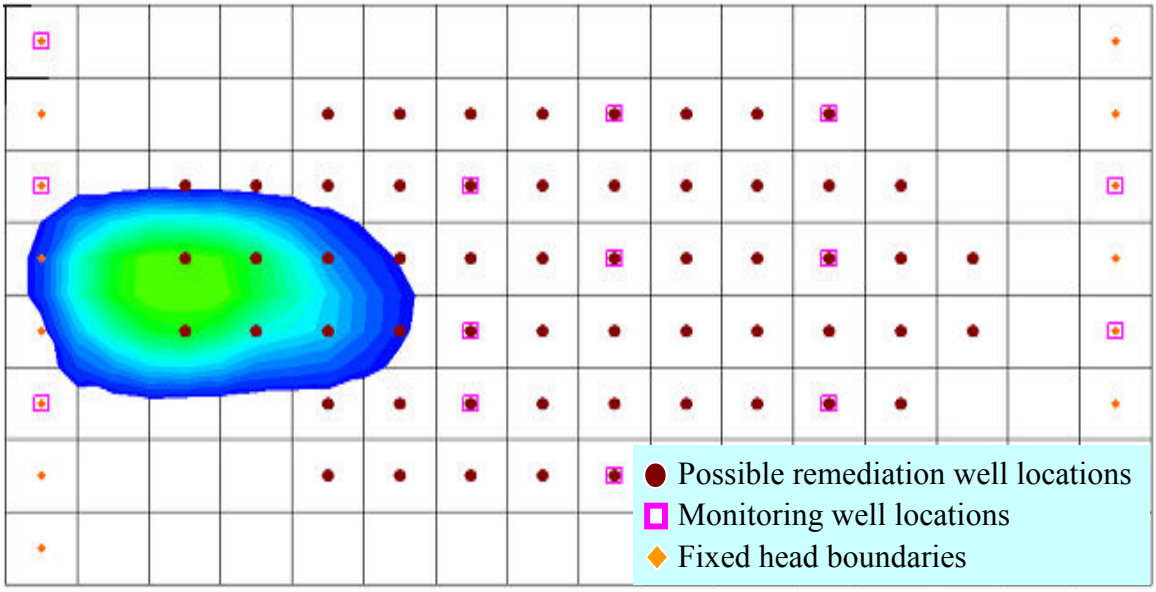

Figure 3.1**.** Plan view of the hypothetical case study.

MODFLOW (McDonald and Haraugh, 1988) and RT3D (Clement, 1997; Clement et al. 1998 and Clement et al. 2000) are used to model the flow field and contaminant concentration in the source area using the finite difference method. The maximum predicted contaminant concentrations in the source area are used by an exposure and risk assessment model to evaluate the individual lifetime human health risk at assumed drinking water wells downgradient from the source area. See MacKay et al. (1986), Freyberg (1986), and Sudicky (1986) and Smalley et al. (2000) for the extensive details.

The objective function representing the total present value cost is minimized in the management model. The pumping well locations and the pumping rates for the injection and extraction wells are the decision variables. The total cost C_{TOT} is,

$$\min C_{TOT} = C_{REM}(x_i, q_i, s_i, u_i, t) + C_{MON}(t) + C_{SYST}(q_i, t) \qquad (3.1)$$

where C_{REM} is the capital and operating costs for the wells; C_{MON} the cost of on-site monitoring; and C_{SYST} is additional capital and operating costs for the ex-situ treatment system.

The decision variable x_i represents the location of well i; q_i is a decision for the pumping rate of well i; s_i is a decision flag determining whether a well is to perform extraction or injection; u_i is a decision flag determining whether well i is installed or not; and t is the length of remediation period.

The objective function shown in equation 3.1 is subject to the following constraints,

$$risk_{t,k}^{TOTAL} \leq TR \qquad (3.2)$$

$$q_{\min,i} \leq |q_i| \leq q_{\max,i} \qquad (3.3)$$

$$h_{\min,i} \leq h_i \leq h_{\max,i} \qquad (3.4)$$

where $risk_{t,k}^{TOTAL}$ is the total individual lifetime health risk at time t and exposure location k; TR is the target total individual lifetime health risk; h_i is the calculated hydraulic head for remediation well i; $q_{\min,i}, q_{\max,i}$ are the minimum and maximum absolute pumping rates for well i; $h_{\min,i}, h_{\max,i}$ are the minimum and maximum allowable hydraulic heads at remediation well i. The constraints on pumping rates were established by setting lower and upper bounds when the pumping rates were discretized. A linear penalty function was used to penalize the cost when the risk and hydraulic head constraints were violated. Thus, the GA actually minimizes the sum of the true cost and the penalty cost,

$$\min\{C_{TOT} + w_1 Viol_{Risk} + w_2 Viol_{head}\} \quad (3.5)$$

where $Viol_{risk}$ and $Viol_{head}$ are risk and head violations with respect to a risk limit of 10^{-5} (or a 1 in 100,000 increased lifetime cancer risk), and an allowable maximum drawdown of 0.12 m. The risk and head penalty weights, w_1 and w_2, were set to 1000 using experimentation to identify the smallest weights that would ensure satisfaction of the constraints (Smalley et al., 2000).

The design objective of the hypothetical case is to minimize the costs associated with pumping well installations and operations so that the human exposure risk due to contaminant migration can be controlled within a specified level. The hypothetical case incorporates flow and transport simulation models but the evaluation of the simulation models is relatively fast. This enables an extensive testing of the methodologies before they are applied to more computationally-intensive case studies.

3.3 Umatilla Groundwater Remediation Case Study

The second remediation case study is a field-scale case study at the Umatilla Chemical Deport at Hermiston, Oregon. The layout of the Umatila depot is shown in Figure 3.2. A brief summary of the case study, which is quite detailed, is given here; for more details see Minsker et al. (2003).

The two major chemical contaminants of concern at this site are RDX (Hexahydro-1,3,5-trinitro-1,3,5-triazine) and TNT (2,4,6-Trinitrotoluene). A pump and treat system is in place for the groundwater operable unit and is shown in Figure 3.2.

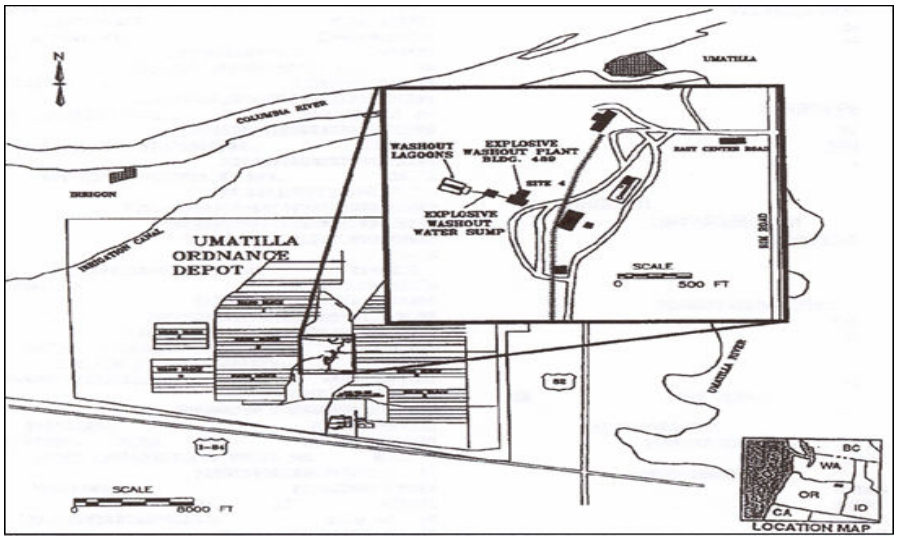

Figure 3.2. Map showing layout of Umatilla chemical depot

The hydrogeology of the site consists of an alluvial aquifer overlying silt and weathered basalt. The Army Corps of Engineers has developed multi-layer two-dimensional flow and transport models (USACE, 1996 and 2000) using MODFLOW (McDonald and Haraugh, 1998) and MT3DMS (Zheng and Wang, 1999). The complete model has 5 layers, 125 rows and 132 columns. Of the 5 layers, the first layer is the alluvial aquifer and the last four are in silt and weathered basalt. The optimization concerns only the alluvial layer because this is the only layer with significant groundwater impact. The conductivity of the layer is heterogeneous, ranging from 375 ft/yr to 1.8x106 ft/yr. The boundary conditions are simulated as constant head along all four edges of the model domain, with net recharge applied to the alluvial layer at a rate of 0.0417 ft/yr.

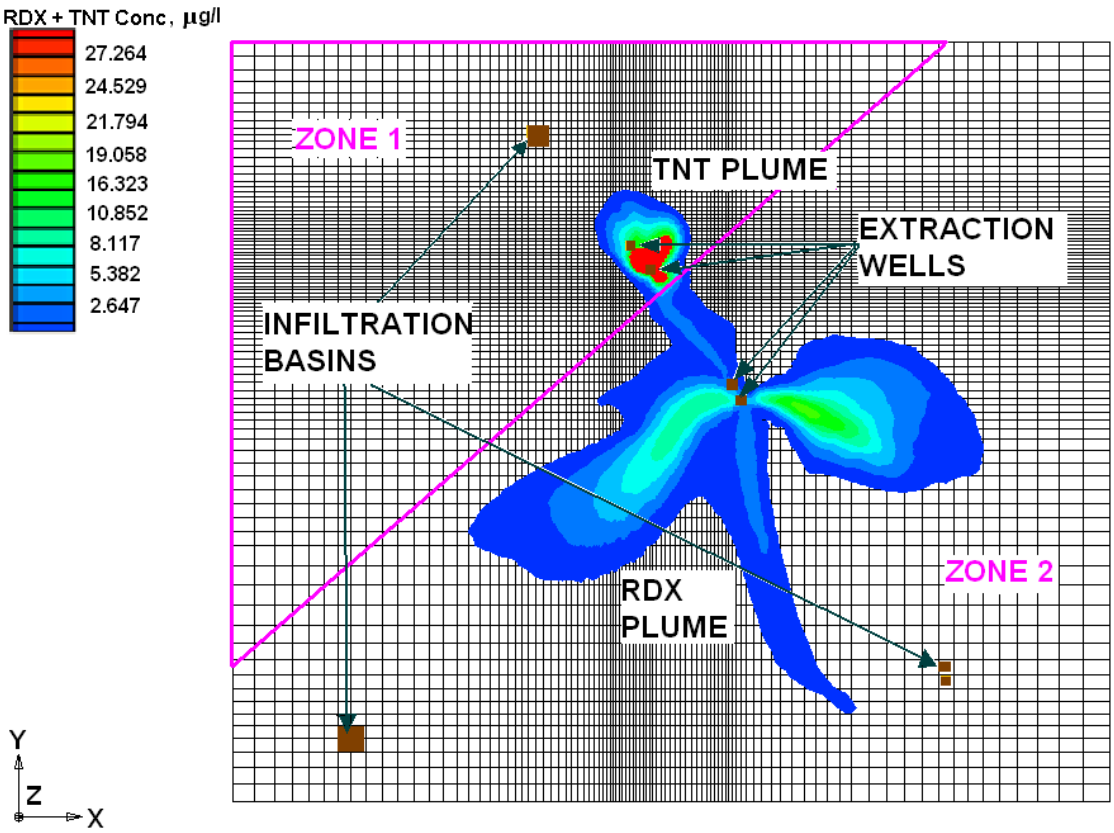

Figure 3.3. Existing pump and treat system and simulated groundwater plumes at Umatilla Chemical Depot (October 2000).

The goal of the optimization is to identify pumping and infiltration rates and locations that minimize the total cost of the pump and treat system, without exceeding the current capacity of the treatment plant, until RDX and TNT are cleaned up. Three optimization formulations were developed by the ESTCP project in collaboration with site personnel. This work focused on formulation one, which identifies locations and pumping rates of extraction wells and injection basins to minimize the associated cost. The original formulation in ESTCP is quite lengthy, but a brief summary is given below:

$$COST = CCW(t,u_{i,nw}) + CCB(t,u_{i,nrb}) + CCG(t,q_i,u_i) + FCL(t) + FCE(t) + VCE(t,q_{i,w},u_{i,w}) + VCG(t,x_i,q_i,u_i) + VCS(t,x_i,q_i,u_i) \quad (3.6)$$

where the decision variable t represents the remediation year; x_i represents decision variables for locations of i^{th} new well/recharge basin; q_i represents decision variables for pumping rates at i^{th} new or existing well/recharge basin; $q_{i,w}$ represents decision variables for pumping rate at i^{th} well; u_i represents binary decision variables for installing/using i^{th} new or existing well/recharge basin; $u_{i,nw}$ represents those binary decision variables for new wells; $u_{i,nrb}$ represents those binary decision variables for new recharge basins; $u_{i,w}$ represents those binary decision variables for both new and old wells. The cost component *CCW* is the capital costs of new wells; *CCB* is the capital costs of new recharge basins; *CCG* is the capital cost of a new granular activated carbon (GAC) unit; *FCL* is the fixed cost of labor; *FCE* is the fixed costs of electricity; *VCE* is the variable electrical costs of operating wells; *VCG* is the variable costs of changing GAC units; and *VCS* is the variable cost of sampling. Constraints for this problem include:

$$\sum_{i \in z_1} q_i + \sum_{j \in z_2} q_j \le 1170 \quad (gpm) \tag{3.7}$$

$$\sum_i q_i^E = \sum_j q_j^R \tag{3.8}$$

$$C_{RDX} \le 2.1 \qquad (\mu g / l) \tag{3.9}$$

$$C_{TNT} \le 2.8 \qquad (\mu g / l) \tag{3.10}$$

$$q_i \le q_{i,\max} \tag{3.11}$$

where z_1 and z_2 are the two different hydrogeological zones shown in Figure 3.3; q_i^E and q_j^R are the pumping rates at extraction well i and recharge basin j; C_{RDX} and C_{TNT} are the maximum RDX and TNT concentrations in the entire region; $q_{i,\max}$ is the maximum pumping rate at a well or recharge basin i. Constraints (3.7), (3.8), (3.9), and (3.10) are

included in the GA's fitness function in the form of linear penalties. Constraint (3.11) is directly included within the binary encoding of the pumping rate decision variables. The overall objective function for this problem then becomes:

$$\min(COST + w_1 Viol_{QE} + w_2 Viol_{RDX} + w_3 Viol_{TNT} + w_4 Viol_Q) \qquad (3.12)$$

where $Viol_{QE}$ is the violation of constraint (3.8); $Viol_Q$ is the violation of constraint (3.7); $Viol_{RDX}$ and $Viol_{TNT}$ are the violations of constraints (3.9) and (3.10). The penalty weights w_1, w_2, w_3, and w_4 were set to 100, 10000, 10000 and 1 respectively, identified using trial-and-error experiments to determine values sufficiently large to satisfy the constraints. Zheng and Wang (2002) and Peralta (2002) found optimal solutions to this case study that achieved the cleanup level within 4 years. To save computational effort, the constraints were modified (from the original cleanup target of 20 years) to require cleanup within 5 years. The modified simulation models ran in about 2 to 3 minutes on a Pentium-4 1.7G personal computer.

CHAPTER 4

AN ADAPTIVE META-MODEL GENETIC ALGORITHM

4.1 Introduction

Chapter 2 discussed the challenges in solving real-world groundwater remediation design problems. The biggest challenge in applying global optimization algorithms is that they require substantially more computational resources than local optimization algorithms. The global optimization algorithms need to evaluate the objective function and constraints repetitively using multiple parameter configurations. This issue is a particular concern for field-scale remediation design problems when the coupled simulations are complex and time-intensive. For example, the last case addressed in the ESTCP transport optimization demonstration required approximately 2 hours for each solution evaluation on a Pentium III desktop computer (Minsker et al., 2003).

Although efficient approximation by meta-models can be used to alleviate the computational burden, most previous applications used offline training to statically train the meta-models. As discussed in Chapter 2, the disadvantage of this approach is that the statically trained meta-models are usually trained on a globally sampled training set, and they may have poor accuracy when the optimization moves into local regions. In the worst case, the errors in the local regions can mislead the optimization algorithm to false optima. The solution to this problem proposed in this work is dynamically trained meta-models. The dynamic meta-models can be updated during the optimization process so that they adapt to local topologies with improved prediction accuracy. This approach was

explored in aerodynamic designs (Jin et al. 2002) but no published applications have appeared in water resources fields.

In this chapter, we develop an adaptive artificial neural network approach within a genetic algorithm framework called adaptive meta-model genetic algorithm (AMGA). The components of AMGA and the methodology are addressed in detail. The algorithm is rigorously tested on the hypothetical remediation case and then the algorithm is applied to the Umatilla case from the ESTCP project (see Chapter 3 for the case study descriptions). Although we focus on the neural-network meta-model and genetic algorithm, the methodology is general enough that it should also be applicable to other approximation models and global optimization algorithms.

4.2 AMGA Components

The section begins with an overview of the AMGA framework, followed by a more detailed description of each component in AMGA. The GA's parameter settings are given in section 4.4 for each case study.

Figure 4.1 shows the flowchart for AMGA, which starts like all simple genetic algorithms. First an initial population of trial designs is randomly generated. Second, the designs are evaluated to determine their fitness with respect to the objectives and constraints of the optimization problem. In this study, fitness evaluations were distributed to multiple computers and cluster systems connected on the Internet. This configuration allowed the simulation models to run in parallel. However, AMGA does not require such

a setup if the computational framework is not available, and the simulation models can be sequentially executed on a single computer.

After the fitness of each design is evaluated, AMGA uses GA operators to create new trial designs for the next iteration ("generation"). This process continues until the population converges to near optimal solutions. In the first few generations, AMGA is in the phase of preparing training sets for the ANN-based meta-models, where designs are evaluated by simulation models only and the solutions constitute the training sets. The chromosomes and corresponding fitness values evaluated by the simulation models are stored in a memory cache for reuse as needed in later generations. Once a sufficient number of simulation model evaluations have been completed to train the ANNs, most of the fitness evaluations are bypassed using ANN evaluations, coupled with the cache of previously evaluated simulations. At each generation, trial designs (adaptively decrease from 20% of the population size to only a few) are sampled for simulation model evaluations according to a sampling policy, and the cache and the ANN retraining sets are progressively updated by the sampled simulation solutions. From time to time, ANNs may be retrained when the retraining criteria are satisfied, which in turn improves their approximation accuracy. The three major components, GA, ANNs and cache, are described in more detail in the following subsections.

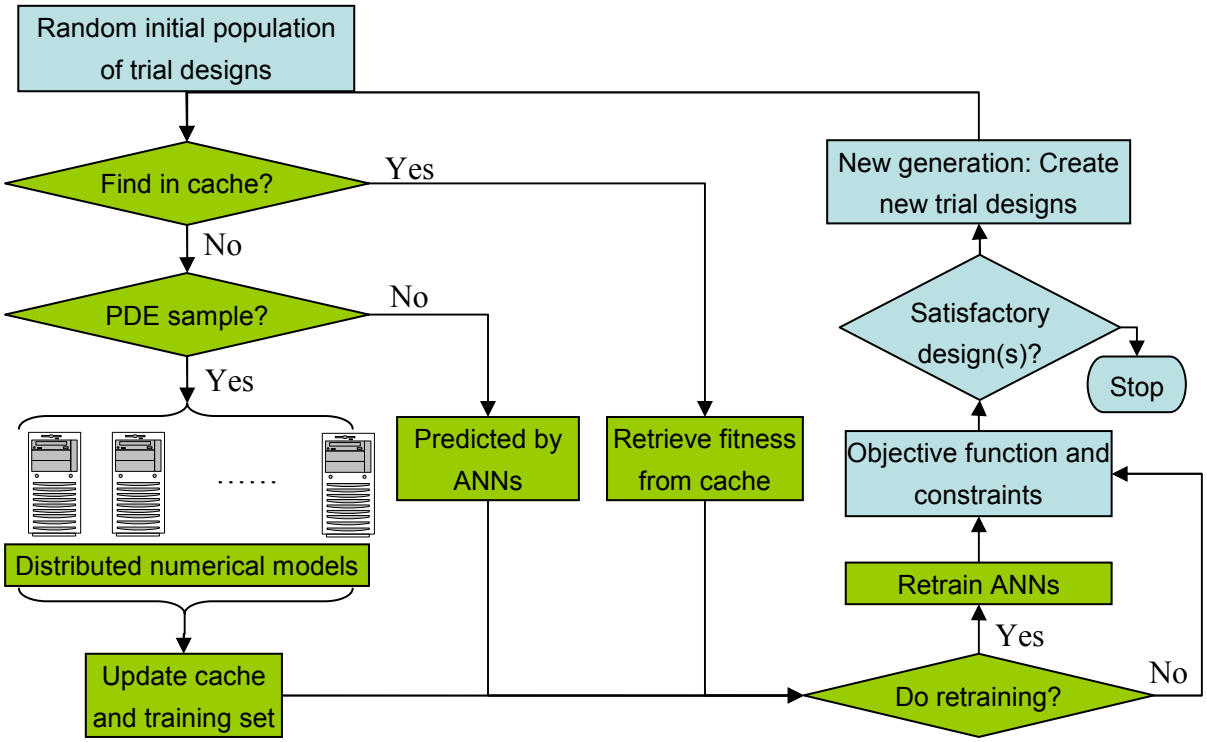

Figure 4.1. AMGA optimization framework.

4.2.1 Genetic Algorithm.

Genetic Algorithms (GAs) are heuristic global searching algorithms introduced by Holland (1975) to simulate the natural evolution process. The algorithm uses chromosomes to encode decision variables, which are string vectors of binary or real numbers representing a particular combination of the decision variables. In the groundwater remediation case, the decision variables are the pumping well locations, their pumping rates, and their operation flags that indicate whether or not the wells are operated or what types of wells are installed (injection or extraction). GAs start from a group or a population of chromosomes, in which each decision variable value is initially randomly assigned. At each generation, GAs decode the chromosomes and then evaluate

their performance using a fitness function representing the objective function and the constraints. In the case of groundwater remediation design, the fitness function is often to minimize cost-plus-penalty for violating remediation goals. GAs then use the three operations of selection, crossover, and mutation to produce a new but presumably better generation (see Goldberg 1989 for details). The evaluation, selection, crossover and mutation operations are repeated until the population converges (which in this study is defined to occur when a specified percentage of the population is the same). Among the four steps, fitness evaluation is the most time-consuming part, which is usually repeated for about 9800 times.

4.2.2 Artificial Neural Networks.

ANNs have been introduced in Chapter 2. In this study, a two-layer, feed-forward neural network was adopted for the meta-models. In this ANN structure, the input data on the input layer are fed without transformation, but with weighting to the hidden layer. After the hidden and output layer transformation, the output layer yields the final predicted results. The activation function of each neural unit is the sigmoid function, which is convenient for derivative calculation within the training algorithms. Details on the specific neural network structures used for each case study are provided in Sections 4.4.1 and 4.4.2.

The backpropagation algorithm is widely used for training ANNs. In this study we used Levenberg-Marquardt backpropagation, which is much faster than the steepest descent backpropagation, has affordable memory requirements, and is less likely to stall

in local minima (Hagan and Menhaj, 1994). Each training episode has a training set and a smaller validation set. The training iterations terminate when the Mean Square Error (MSE) on the validation set stops decreasing, which avoids overfitting and requires fewer training iterations.

4.2.3 Caching.

Caching is a technique of temporarily storing the simulation-model-evaluated chromosomes and their fitness values into a memory space so that the evaluated chromosome and fitness information can be searched and retrieved quickly later on. Note that meta-model-evaluated chromosomes are not cached since the meta-model evaluations are efficient. Caching can improve the GA's performance by using this storing and retrieving technique instead of going through the numerical model evaluations when the evaluated chromosomes reappear (Kratica, 1999). Figure 4.2 shows a test run result using SGA and cache on the hypothetical case study introduced in Chapter 2. The figure shows the proportions of numerical model (PDE) evaluations and cache evaluations in each generation during a complete SGA+CACHE run. As shown in the figure, in early generations, when the cache is small and the population is diverse, most, if not all, of the chromosomes have to be evaluated by the numerical models and the results are added into the cache. As the GA's population becomes more and more homogenous in later generations, more and more previously tested chromosomes will appear and those evaluations can be bypassed by determining their fitness values in the cache.

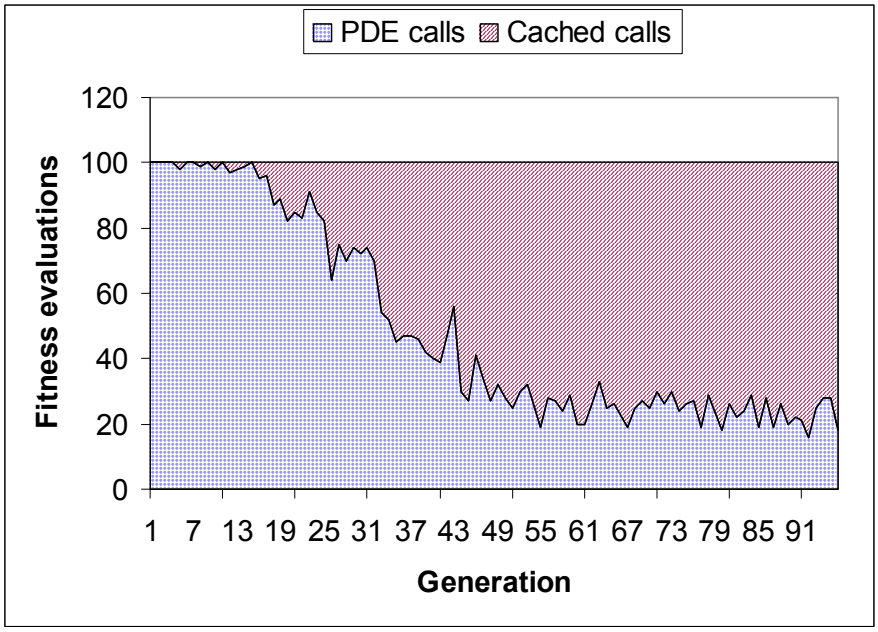

Figure 4.2 Cached fitness evaluations in an SGA+Cache run

Caching plays a more active role when incorporated into AMGA, which relies on meta-model models to estimate the fitness values. Fitness approximation from meta-models will inevitably introduce searching error because fitness is the only information that the GAs use to identify the searching direction. When the same chromosomes appear repeatedly, AMGA can use the cached true fitness instead of the meta-models' estimation to improve searching accuracy. The caching technique is most beneficial when the GA's population is nearly homogeneous in a local region, which occurs as the population approaches convergence. At that stage, a large amount of the GA's searching information is based on the cached true fitness values rather than estimations, so caching improves local searching accuracy in addition to saving fitness evaluations.

A caching algorithm must be able to determine a chromosome's fitness efficiently. Sequentially searching through the cache requires a time complexity proportional to the

size of the cache. AMGA implements the caching algorithm using AVL tree (Lewis and Denenberg, 1997), which has been standardized in many libraries. The algorithm uses a balanced binary search tree with an upper bound of $O[Log(Nc)]$ for the search time complexity, where Nc is the number of chromosomes stored in the cache. Figure 4.3 shows the diagram of an AVL tree structure. Each node in the AVL tree contains a chromosome and the associated fitness value. The chromosomes are used as the key to compare two different nodes. In the AVL implementation, this is achieved by comparing the string vectors of the two chromosomes bit by bit. Each node may have a left child node and a right child node. All the child nodes on the left side (left sub-tree) of the current node are smaller the current one, and the child nodes on the right side (right sub-tree) are bigger than the current node. This structure is carefully maintained so that the nodes (except those at the bottom level) always have two children (a complete tree). At each level, the number of nodes is doubled from the previous level, so the heights of the bottom nodes are bounded by $Log_2(Nc)$. This design guarantees that the insertion, deletion, and searching operations traverse at most $Log_2(Nc)$ nodes; therefore the maximum time complexity is bounded by $O[Log(Nc)]$, which is very efficient.

More efficient caching algorithms, such as hash table (Lewis and Denenberg, 1997), may be adopted to improve the caching efficiency. However, as the GA's time complexity for water resources optimization is typically dominated by the underlying simulation models' execution time (as in the cases examined in this research), the overall performance improvement would likely be insignificant.

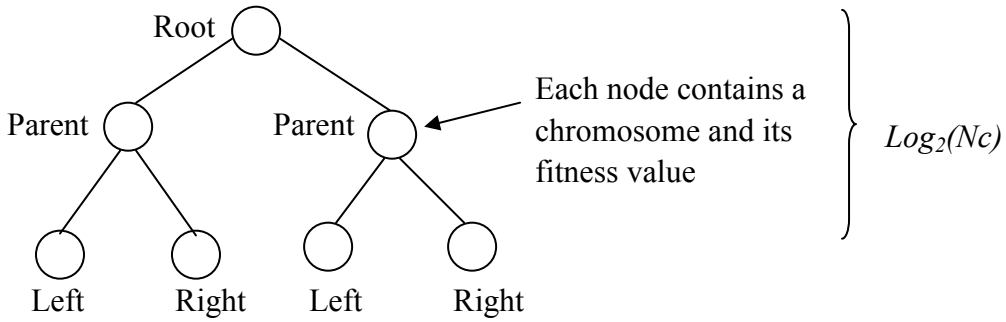

Figure 4.3. AVL tree cache

The memory requirement of caching is on the order of tens of kilobytes for typical groundwater remediation problems with chromosomes having hundred of bits (assuming 50% of the individuals are cached). Such a memory requirement is not a problem for modern PCs, which usually have hundreds of megabytes of physical memory.

4.2.4 Distributed Computing Infrastructure

Parallel computing is an important approach to reducing computational time for time-intensive applications. GAs are naturally suitable for parallel computing because the fitness evaluations in each generation have no internal dependencies. By enabling fitness evaluations to be performed for the entire population simultaneously, parallel computing can significantly improve GAs' performance. A popular approach of running GAs in parallel is using existing parallel computing libraries, such as the MPI programming interface. However, the GAs linked with MPIs usually have to be deployed on mainframe computers or cluster systems. The disadvantage of this deployment is that mainframe computers are often expensive and less accessible to researchers and engineers. Moreover, since multiple users compete for the limited computing resources, the available computing nodes to each user are often limited. The applications often have to wait in a

queue before the computing resources are allocated, which makes it difficult for time critical applications.

A new computing technique that has been enabled by the significant increase of computing power is the distributed computing approach. The distributed computing approach does not rely on expensive mainframe computers. Instead, the computing jobs can be distributed through the high speed internet to interconnected computing nodes. An early application of the distributed computing approach in Civil and Environment Engineering is Balla and Lingireddy (2000), who developed a GA model that can run on personal computers connected by a local office network. This chapter presents a hierarchical distributed computing infrastructure that can effectively deploy computing jobs to personal or office computers, cluster systems, as well as mainframe computers.

Figure 4.4 shows the infrastructure of the proposed distributed computing system. The system is comprised of a GA machine, multiple proxy nodes and multiple service nodes. The GA machine hosts the optimization algorithm, the proxy nodes are intermediate nodes that manage the actual service nodes, and the service nodes are the physical nodes for running simulation models. The three components are organized in a hierarchical relationship with the GA machine at the top level. The connection lines in the figure are communication channels. Messages and commands are sent back and forth through the communication channels so that jobs are delivered to the proxy and service nodes, and results are collected back to the GA machine. The communication channels are established using the TCP/IP protocol, which can be encrypted if the channels across

several local networks. The tokens shown in the figure represent the available CPU resources. The service nodes detect the idle CPUs and generate a token for each idle CPU. The GA machine must acquire a token before it emits a job to a service node. After the job is finished, the token is returned to the service node and it is available for other job requests. Note that the tokens contain CPU speed and memory space information so that simulation jobs can be scheduled appropriately according to the resource requirements.

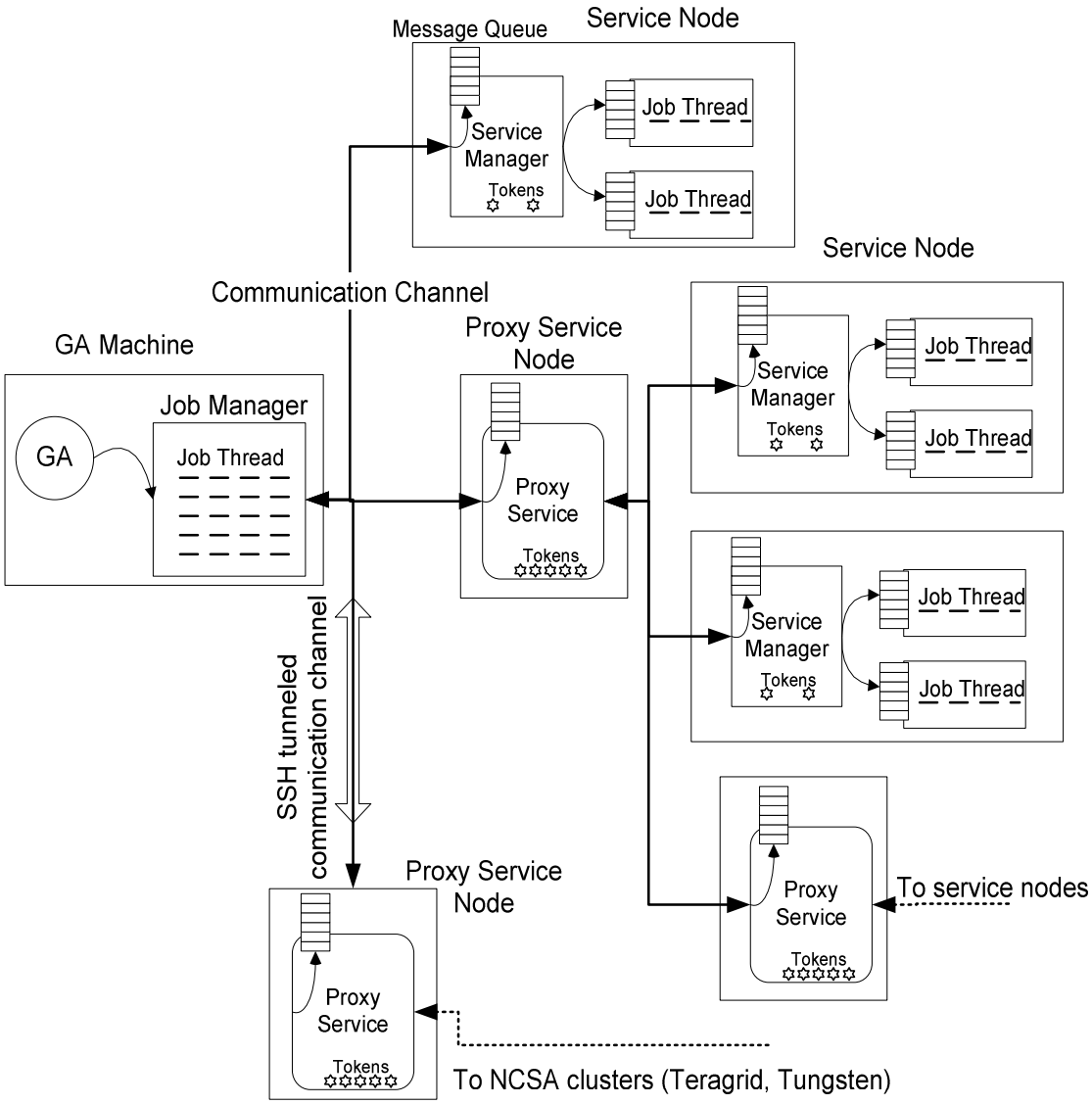

Figure 4.4. The distributed computing infrastructure diagram

The GA machine is usually a desktop computer in which the optimization algorithm is executed. The GA machine has a job manager to coordinate and schedule the simulation jobs. In each generation, after the GA generates the simulation jobs, it asks the job manager to distribute the jobs to the server or proxy nodes. The job manager establishes communication channels with the service nodes or proxy nodes to request CPU tokens. For each acquired token, the job manager creates a job thread and forwards the job to the thread. The job threads are executed concurrently. They send the instructions to the appropriate service or proxy nodes so that data files are exchanged and simulation jobs are executed.

The service nodes are the terminal nodes that actually run the simulation jobs. A service node may have multiple CPUs. The service node generates a token for each CPU. If multiple job requests are received from the GA machine, the service node starts multiple jobs and runs them concurrently. Like the job manager in the GA machine, a service node also has a service manager to coordinate the running simulation jobs. For each scheduled simulation job, the service manager creates a job thread and attaches it to a CPU resource. The job thread then starts the job process and monitors its progress. In addition, the service manager has a message queue that collects the messages or instructions received from the connected communication channel. The service manager is responsible for dispatching the messages or instructions to the job thread queues, where the job threads interpret the instructions and respond accordingly.

The last component is the proxy nodes. The proxy nodes are the key to the hierarchical structure of the distributed computing system because they may connect with multiple service nodes or proxy nodes at lower levels. The proxy nodes don't actually run simulation jobs. Instead, they are like switches in a network that forward messages sent between higher level nodes (a higher level proxy node or the GA machine) and lower level nodes. To the lower level nodes, the proxy nodes are like token managers. They collect tokens from lower level nodes and forward jobs to them. But to the higher level proxy node or the GA machine, a proxy node is like a virtual service node with a large amount of CPU tokens. Introducing the proxies into the system has two benefits. First it saves communication channels. The service proxies are usually located within the same local network as the managed service nodes. The GA machine needs to establish only one communication channel with each proxy node, no matter how many actual service nodes are managed by the proxy node. If the proxy node and its lower level service nodes live in a remotely connected cluster system, this structure can significantly decrease the communication overhead because otherwise the GA machine would have to connect each of the remote service nodes. Second, this structure can incorporate security mechanisms easily because there is only one communication channel to be protected for each remotely connected cluster. Currently, the SSH tunneling technique is used to encrypt the communications if the service or proxy nodes are remotely connected. SSH is an industrial standard that provides strong authentication and secure communications over unsecured TCP/IP channels. The SSH tunneling technique first establishes a secure SSH

channel with a remote computer. Then it disables the other unsecured communication channels. The technique encrypts message packages and forwards them to the destination computer through the SSH channel.

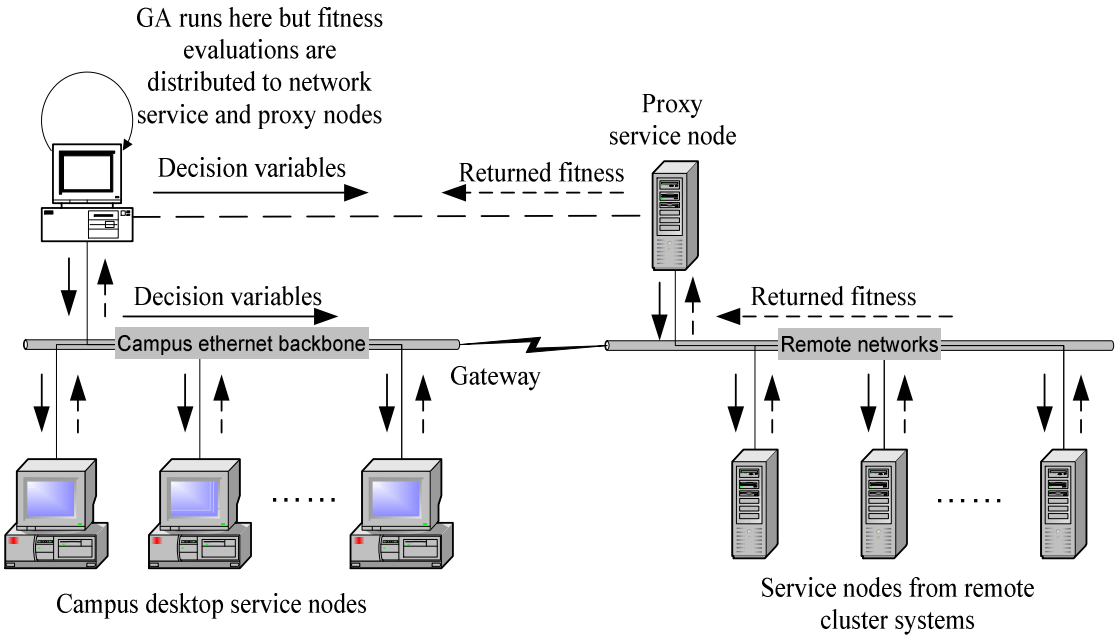

Figure 4.5. The distributed computing infrastructure deployment diagram

The distributed computing infrastructure was deployed on office and campus desktop computers, and several cluster systems in the National Center for Supercomputing Applications (NCSA). Figure 4.5 shows the deployment diagram. In this deployment, the GA directly connects the office and campus service nodes since they are within in the same campus local network. The service nodes in NCSA clusters (Teragrid, Tungsten and Tgc) are managed by their proxy service nodes, which are connected by the GA machine though the campus gateway. The system follows a communication protocol so that the running GA and the distributed jobs are coordinated. A brief description of the system protocol is as follows. First, when the GA starts, the GA machine and the service

or proxy nodes do “handshakes.” During the handshake period, the service nodes communicate their specifications (eg. CPU speed, memory space) to the GA machine and then exchange the necessary data files. Second, at each generation, the GA job manager schedules the simulation evaluations to the service (proxy) nodes with higher priorities given to faster nodes. The job manager enters a waiting loop after all of the simulations have been scheduled. Finally, when the service nodes finish the simulations, the job manager retrieves the results (fitness values) and wakes up the GA. Then the GA moves to the next generation. Any computer can be configured as a service or proxy node as long as it runs a service program. The service program runs in the background and monitors the system. It makes effective use of idle computers without interfering with other users by using only idle CPU time for simulation runs (eg. when the computer is logged off, locked, or left unattended for a long period). The system deployed on office computers obviously provides more CPU power at night and on weekends when almost all of the computers are idle.

Compared with other parallel approaches, the distributed system is flexible and scalable – it can run on desktop computers and/or cluster systems; it can run across multiple operating systems (Windows, Linux and Unix); it is inexpensive – existing desktop computers can be easily configured as service nodes to share their idle CPU time and no special hardware equipments are required; it can be easily deployed – only a service program should be started, the system can have unlimited service nodes at local

or remote sites, and nodes can be added or removed at any time without stopping the running GA.

The distributed computing infrastructure was developed in C++ language using TCP/IP protocol for network communication. Currently the infrastructure provides a peak of about 80 service nodes.

4.3 AMGA Implementation

In the AMGA framework shown in Figure 4.1, the ANNs are adaptively retrained to improve their prediction accuracy. In this section, a benchmark problem is used to illustrate how the performance of ANNs is improved after retraining. Then fitness sampling issues are discussed in detail and several approaches are proposed for the ANN training and retraining.

4.3.1 ANN Approximation Properties.

ANNs are popular tools for approximating function response surfaces. Given a response function surface defined by a finite number of training points, an ANN can find infinite number of prediction functions that interpolate the training points. In reality, however, interpolating the training points often leads to overfitting. Therefore, the ANN training algorithms usually search smooth response functions that fit the training points to avoid overfitting. If the actual function has many local minima, the trained response surface based on globally sampled training points may smooth out local details to maintain a small Mean Square Error (MSE) in the global region. This globally smoothed

response function may have high errors in local regions, where global minima may be found. The property is illustrated using a two dimensional Ackley function. The formulation of the two dimensional Ackley function is,

$$f(x) = 20 + e - 20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{n}\sum x_i^2}\right) - \exp\left(-\frac{1}{n}\sum\cos(2\pi x_i\right) \quad (4.1)$$

- where n is equal to 2

Assume the global region is the region covered by [-4,4] for both x and y. Figure 4.6 shows the global overview of the true function surface of the Ackley function. The function has many local regions, and the global minimum is at the origin (0,0). An ANN was trained to approximate the Ackley function using 150 points sampled within the global region. The ANN had 2 inputs for x and y coordinates, 8 nodes in the hidden layer and one output. Figure 4.7 shows the ANN's response surface.

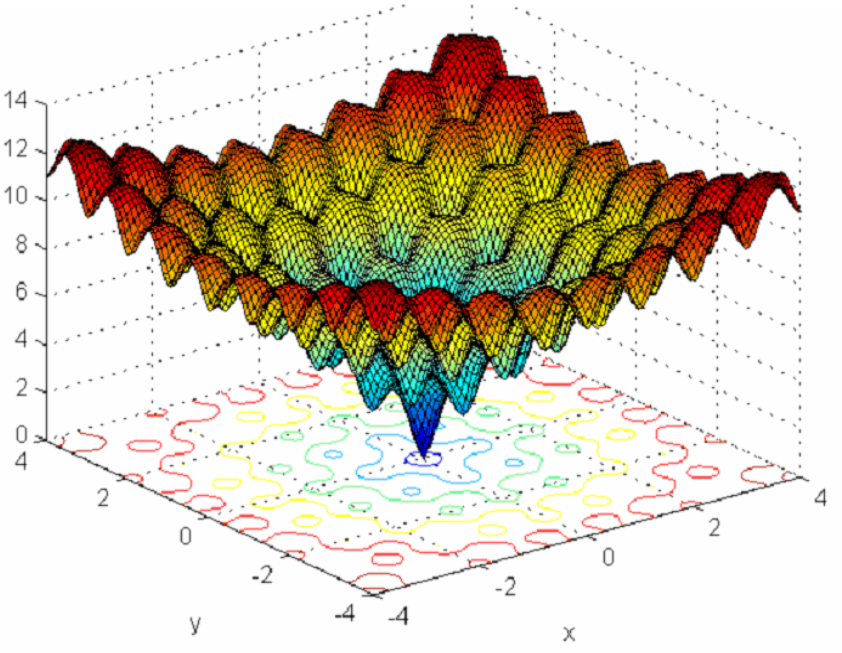

Figure 4.6. Global overview of Ackley function in [-4, 4]

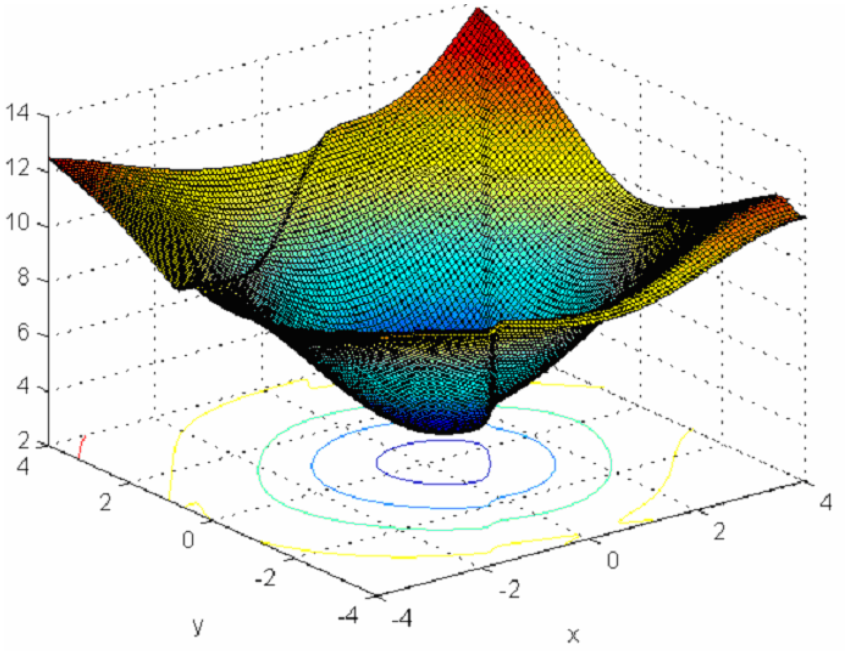

Figure 4.7. Global overview of ANN's approximation for Ackley

The ANN's response surface correctly approximates the general shape of the Ackley function. If the ANN were used to replace the Ackley function in a GA framework, the GA's population would quickly converge to the local region within [-1, 1] because the meta-model captures the backbone of the surface despite errors in local regions. After the GA finds the right local region, however, it is likely to converge to a false minimum. Figures 4.8 and 4.9 show the local details of the Ackley function and the ANN's response surface. While the true global minimum is at the origin as shown in Figure 4.8, the global minimum in the ANN's response function misses it as shown in Figure 4.9. This inconsistency explains why GAs may fail when using statically trained ANNs as meta-models.

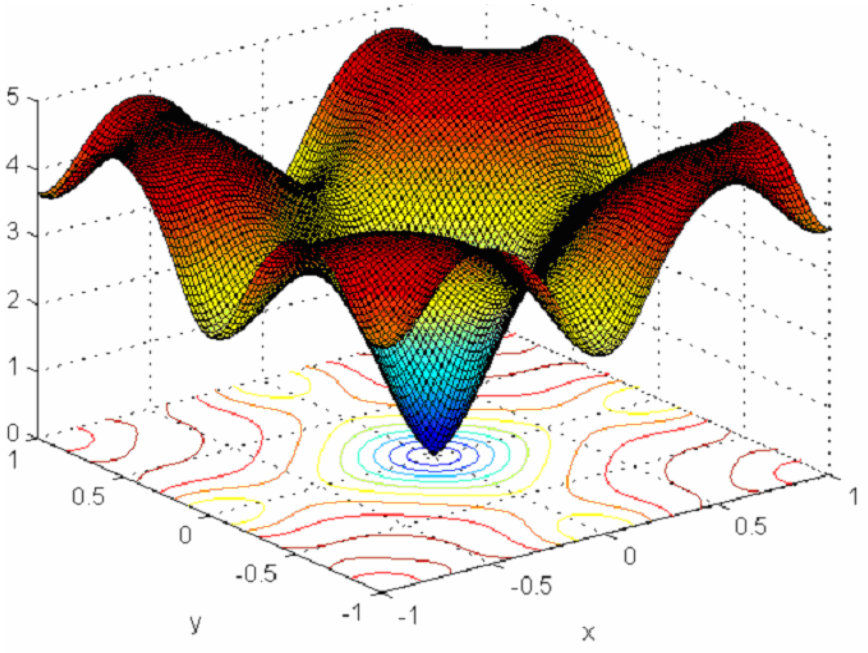

Figure 4.8. Ackley function local detail in [-1, 1]

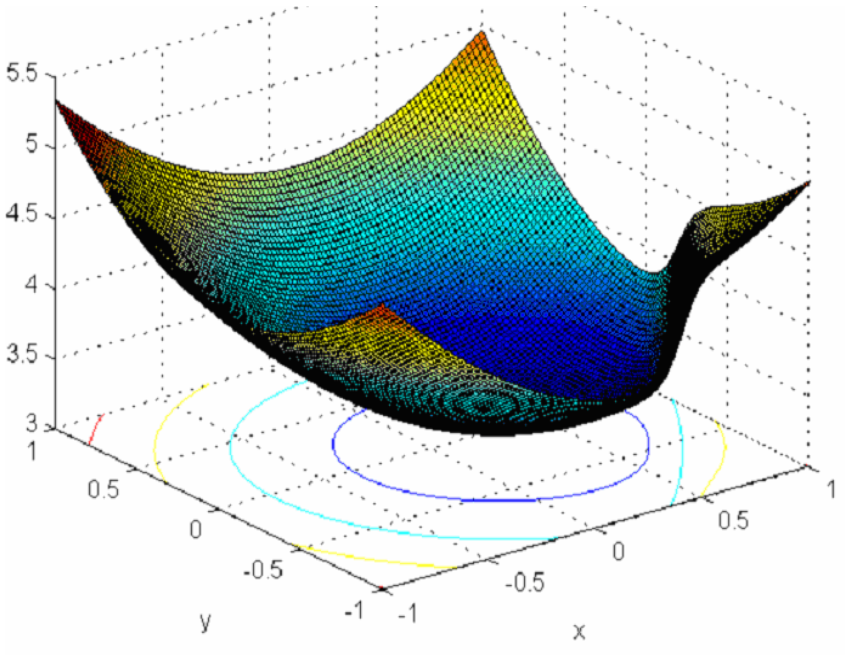

Figure 4.9. Local details of ANN's approximation in [-1, 1]

Increasing training points by random sampling can alleviate the problem for a simple surface, but for a complex surface like the Ackley function, it is very inefficient. A better approach to improving local approximations is increasing local sampling in the training set. If we sample 100 of the 150 points from the local region [-1, 1], and leave the other 50 points untouched, then the local approximation of the same ANN can be substantially improved. Figure 4.10 shows the retrained response surface in the global region. The retrained ANN still captures the general function shape. In the local region of [-1,1], the retrained ANN is able to approximate the local topography and captures the global minimum point with high accuracy (Figure 4.11).

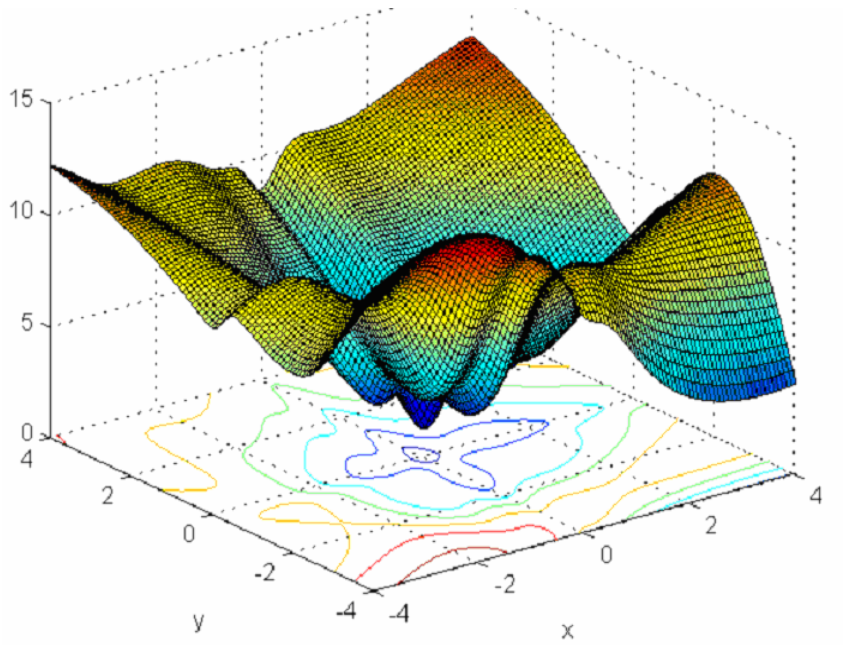

Figure 4.10. Retrained ANN response surface (100 of the 150 points from local sampling)

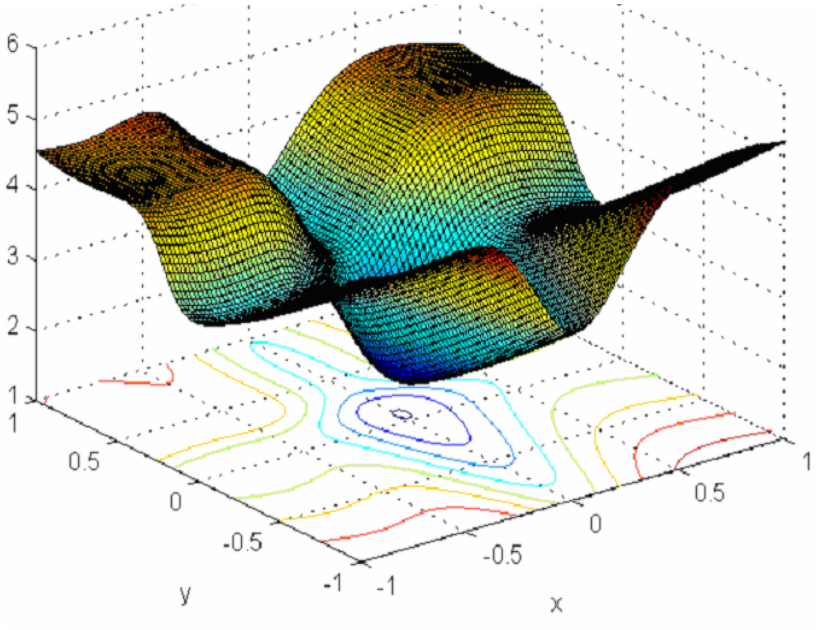

Figure 4.11. Improved local approximation of the retrained ANN

The property that an ANN can have global or local approximation ability by changing the sampling proportion of the training set is beneficial when applied to a GA framework. In early generations, it is more critical to identify the promising regions, so a random sampling in the global region should suffice for training the ANNs. After the population moves into local regions, adaptive sampling in these regions can produce local training points, which are then used to update the response surface to improve the identification of local topographies and locations of minima. In a GA framework, sampling training points should be adaptive. In early generations, when GAs are moving quickly from region to region, more sampling and retraining are required. In later generations, less sampling is required because the local regions become relatively smooth

as the population converges to near optimal solutions, hence the ANNs are probably sufficiently accurate.

4.3.2 Fitness Sampling

To implement adaptive sampling, the true fitness functions (in the case of water resources optimization, usually simulation models that solve PDEs) are sampled periodically. Two factors must be considered to ensure effective and efficient fitness sampling: sampling rate and sampling selection strategy. These factors are discussed in more detail below.

Sampling Rate. Sampling rate determines how many individuals should be sampled from a population in each generation (i.e., how many individuals are evaluated with the true fitness function). A heterogeneous population needs a higher sampling rate to improve GA's global searching ability and increase the retraining frequency (assuming a fixed number of sampling needed for each retraining). A homogeneous population needs much less sampling because the ANNs are already adapted to the homogeneous local regions. In order to avoid redundant sampling caused by a fixed sampling rate, AMGA uses an adaptive strategy that can automatically decrease the sampling rate by tracking the homogeneity of the current population.

A scaled averaged standard deviation (*SD*) of the input variables (decision variables) to the ANN is used to estimate the homogeneity of a population. Each input variable is scaled to the range [-1.0, 1.0], making the method problem independent. The standard deviations of the scaled input variables are calculated, and these standard

deviations are averaged. *SD* changes from a value usually close to 0.58 (the standard deviation of the uniform distribution in the range of [-1.0, 1.0]) in the first generation to a value close to 0 when converging. This value is then smoothed out to make the estimation smoother in a dynamic environment. The formulas for *SD* and the smoothed *SD (SSD)* are given below:

$$SD_i = \frac{1}{n}\sum_{k=1}^{n} std(X_k) \tag{4.2}$$

$$SSD_i = SSD_{i-1} + \eta(SD_i - SSD_{i-1}) \tag{4.3}$$

X_k — vector of scaled input variable k

n — number of input variables (decision variables)

SSD_i, SD_i — (smoothed) scaled averaged standard deviation at generation i

η — smoothing constant in [0, 1], eg. 0.5.

std — function to compute the standard deviation of vector X_k

Using these terms, the sampling rate used in this work is then:

$$S_i = \frac{SSD_i}{SSD_0} S_0 \tag{4.4}$$

where S_i is the sampling rate and S_0 is the initial sampling rate. The number of sampled individuals in a population in each generation is then calculated as:

$$n_i = S_i N \tag{4.5}$$

where N is the population size.

The values of *SD* and *SSD* during an AMGA run are shown in Figure 4.12. As shown in the figure, the *SD* and *SSD* decrease as the generation increases. The values decrease faster in early generations as the populations are rapidly changing. After about 60 generations, the population becomes homogeneous, and the *SD* and *SSD* values

become stable. Because *SSD* is a moving average of *SD*, the *SD* curve is smoother than the *SSD* curve.

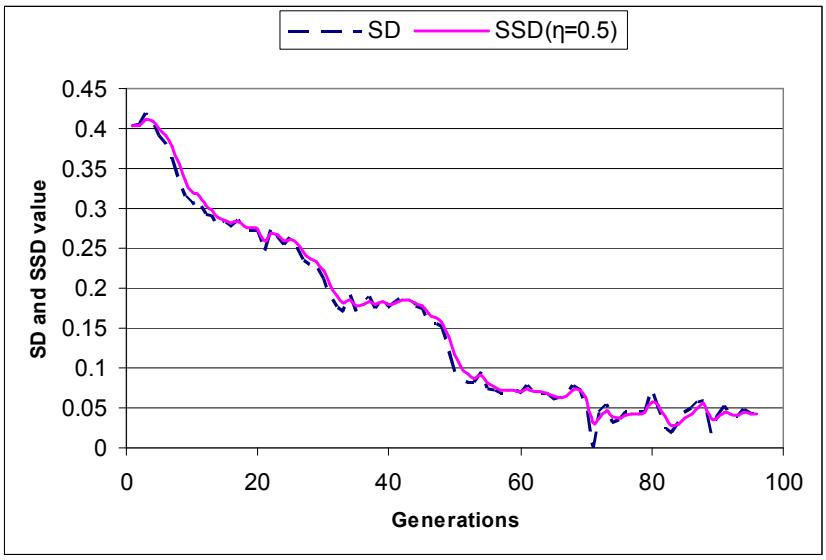

Figure 4.12. SD and SSD in an AMGA run

The sampled solutions are stored in a retraining pool of a user-specified size (e.g., 100 solutions). When the retraining pool is full, retraining is triggered and the pool is emptied for the next retraining. In this approach, S_0 is the parameter that determines AMGA's overall sampling level. Its impact on AMGA's performance will be addressed in the results section.

Sampling Selection Strategy. Sampling selection strategy determines which individuals should be sampled from a population, given the current sampling rate. The most straightforward strategy is *random sampling*, where any chromosome in the population has an equal likelihood of being sampled. Random sampling is a conservative method because it does not use any knowledge from the existing ANNs. The most informative yet aggressive method is called *best sampling*. This method sorts the

individuals by their ANN estimated fitness and then samples the best individuals for simulation model evaluations. Because the ANN's prediction is not error-free, best sampling can be misled by erroneous knowledge at times. The third method, which uses information from the ANNs in a less aggressive manner, is called *tournament sampling*. Tournament sampling is similar to tournament selection, where small groups in the population are randomly selected and the chromosome with the best ANN fitness estimation is chosen for fitness sampling. Finally, the last approach is a combined *tournament+best sampling* method, which uses tournament sampling to increase sampling coverage for the first half of the run and then switches to the best sampling method for the second half of the run, when the population is converging to near optimal solutions and accurate fitness evaluations of the best members are most useful.

4.3.3 ANN Training and Retraining

The fitness sampling described in the previous section periodically triggers retraining of the ANNs. This section discusses several issues that must be considered in training and retraining, including the number of initial training generations, retraining method, and retraining frequency.

Initial Training Generations. All of the individuals in the first several generations of AMGA must be evaluated by the simulation models to generate the ANN training set. The number of initial training generations (G_0) determines how long this preprocessing stage lasts. Preprocessing is time consuming due to the number of simulation model calls, so a smaller G_0 is desired. Different G_0 levels are tested in the results section.

Retraining Set Management. As more and more sampled solutions are generated from simulation model evaluations, the retraining set must be managed. Two approaches are tested in this study. The first one is termed the *growing set* approach, in which the sampled solutions combine with the old training set to form a new training set. The second approach keeps the size of the training set constant by discarding the oldest training solutions when new solutions are added (*fixed set* approach).

Retraining Method. When the ANNs need to be retrained, the training algorithm can either load the previously trained weights and continue the training episodes on the new training set, or it can re-initialize the ANN weights to random values and completely retrain the ANNs. Although the first approach is faster, the ANNs often remain trapped in locally minimal weights that are no longer relevant (Zhang, 1994). Given that the training time for most complex applications is relatively small compared to the duration of the GA run, the second retraining approach is used here to promote accuracy.

Retraining frequency. Retraining frequency determines when the ANNs should be updated during an AMGA run. Retraining frequency should decrease in later generations as the search progresses into relatively smoother local regions. AMGA addresses this need by allocating a fixed retraining pool and putting new solutions into the retraining pool whenever they are generated by the sampling. Once the retraining pool is full, retraining occurs and the pool is emptied for the next retraining. With this approach, as the adaptive sampling rate (calculated using Equations 4.2 – 4.5) decreases in subsequent generations, the retraining frequency follows the same trend.

4.4 Test Cases and AMGA Setups

The groundwater remediation test cases presented in Chapter 3 were used to test AMGA's performance. The hypothetical groundwater remediation case can be solved quickly, hence AMGA was extensively tested with the hypothetical case to identify good parameter configurations. Then AMGA was applied to the Umatilla case study using only the best parameter settings identified in the hypothetical case study to reduce computational demands. The setups of AMGA for the two test cases are summarized in the following sub-subsections.

4.4.1 AMGA Setups for the Hypothetical Case

The hypothetical groundwater remediation case study was previously solved using a simple GA by Smalley et al. (2000). The GA solving this problem had a population of 100 individuals, the crossover probability was 0.5, and the mutation probability was 0.002. These parameters were determined using the approach developed by Reed et al. (2000) in previous work. The fitness evaluation of each individual requires about 2 seconds (on a Pentium-4 1.7G desktop computer) using the simulation models. The algorithm converges after about 96 generations. The optimal solutions found by the GA have a total present value cost of approximately $186,000 with a interest rate of 0.1.

AMGA used the same GA parameters but with two ANNs incorporated. The ANNs are trained to predict hydraulic heads and human health risks needed for the optimization constraints. The first ANN, replacing the groundwater flow model, uses the pumping well locations and pumping rates as inputs to predict the hydraulic heads at the

pumping wells. The pumping well locations, the pumping rates, and the hydraulic heads at those wells are then used to train a second ANN. The second ANN predicts the maximum human health risk at all monitoring wells, thereby replacing the contaminant transport model and the risk assessment module. The log of the risk is used to avoid scaling problems associated with extremely small numbers.

Table 4.1. ANN structures of the hypothetical case

	ANN-1	ANN-2
Inputs	$x_1, x_2, x_3, y_1, y_2, y_3, q_1, q_2, q_3$	$x_1, x_2, x_3, y_1, y_2, y_3, q_1, q_2, q_3, h_1, h_2, h_3$
Hidden neurons	12	14
Outputs	h_1, h_2, h_3	*log-risk*

where (x_i, y_i) is the coordinate of pumping well i, q_i is the pumping rate of well i; h_i is the hydraulic head of pumping well i; and *log-risk* is the log of the total risk (*TR*) in equation 3.2.

The ANNs' hidden layer structures were determined using trial and error experimentation. The nodes in the hidden layer were gradually increased until the improvement in the ANNs' performance was minimal. The ANN structures are listed in Table 4.1. The groundwater flow ANN has 9 units (pumping well locations and pumping rates) in the input layer, 12 units in the hidden layer and 3 units (hydraulic heads at the pumping wells) in the output layer. The risk evaluation ANN has 12 (pumping well locations, pumping rates and hydraulic heads at the pumping wells) units in the input layer, 14 units in the hidden layer and 1 unit (log of risk) in the output layer. Inputs and outputs of the ANNs are scaled to the range [-1.0, 1.0]. Mean Square Error (MSE)

between the predictions and the observations is used to evaluate the predictive accuracy. All of the MSE results presented here are based on the normalized data.

4.4.2 AMGA Setups for the Umatilla Case

The Umatilla case study was also previously solved by simple GA using parameters selected with the approach developed by Reed et al. (2000). A population size of 160, a crossover probability of 0.5 and a mutation probability of 0.00625 are used. The GA requires 60-80 generations before converging, depending on the random seed. The optimal design solutions found by the GA has a cost of approximately 1.66 million dollars.

Two ANNs in AMGA are used to substitute for the flow and transport models. The structures of the two ANNs are listed in Table 4.2. The first ANN uses the pumping well locations (x and y coordinates) and the pumping rates of the pumping wells and injection rates for the recharge basins as inputs. It predicts the maximum concentrations of RDX and TNT in the fifth year, as well as the variable GAC cost component (VCG, equation 3.6), which depends on the source area concentrations. The ANN has 28 inputs, 12 nodes in the hidden layer, and three outputs. The second ANN has the same inputs, 14 nodes in the hidden layer, and two outputs for predicting the ratios of the maximum concentrations of RDX and TNT in the fifth year to their values in the fourth year. The rationale for predicting the ratios rather than the maximum concentrations (needed for the cleanup constraints) is that modeled contaminant concentrations usually decrease over time during a cleanup operation. Directly predicting the concentrations in the fourth year

sometimes produces a contradictory trend, with concentrations increasing over time. The normalized ratio scaled in the range of [0,1] implicitly forces the neural networks to follow the correct trend.

Table 4.2. ANN structures of the Umatilla case

	ANN-1	ANN-2
Inputs	$x_1, x_2, x_3, x_4, x_5, x_6, x_7, y_1, y_2, y_3, y_4, y_5, y_6, y_7, q_{1,nw}, q_{2,nw}, q_{3,nw}, q_{4,nw}, q_{5,nw}, q_{6,nw}, q_{7,nw}, q_{1,ow}, q_{2,ow}, q_{3,ow}, q_{4,ow}, q_{5,ow}, q_{6,ow}, q_{7,ow}$	same as ANN-1
Hidden neurons	12	14
Outputs	$C_{RDX(5)}, C_{TNT(5)}, VCG$	$C_{RDX(5)}/ C_{RDX(4)}, C_{TNT(5)} / C_{TNT(4)}$

where (x_i, y_i) is the coordinate of new pumping well i; $q_{i,nw}$ is the pumping rate of new well i; $q_{i,ow}$ is the pumping rate of old well i; $C_{RDX(i)}$ and $C_{TNT(i)}$ are the maximum concentrations of RDX and TNT at the ith year; and VCG is the variable GAC cost component from equation 3.6.

4.5 Results

This section presents the results of applying AMGA to the above two test cases. First we show how the static training approach can fail for the hypothetical test case. Then AMGA's performance is explored in detail with respect to different parameter configurations. Finally, the best performing AMGA is applied to the Umatilla test case and the results are discussed.

4.5.1 Applying AMGA to the Hypothetical Case

We first demonstrate the importance of adaptive retraining by solving the hypothetical case with a standard GA coupled with statically trained ANNs using an offline method (no model updating). Then AMGA is rigorously tested on the hypothetical test case with different parameter configurations. Finally, the results of applying AMGA and a standard GA are compared.

Problems of Static Training

In section 4.3, we noted that statically trained ANNs can lead GAs to false optima because of decreasing prediction performance. This phenomenon was examined using a standard GA coupled with statically trained ANNs. The solutions from the first eight generations of the GA were evaluated by the coupled numerical models, and then the solutions were used to train the neural networks. During the following optimization process, the ANNs replaced the numerical models without any subsequent updating (a standard ANN+GA). The cache was disabled to ensure that the results address only the effects of ANN updating. In each generation, the ANNs' performance was also tested offline by evaluating the predicted solutions using the numerical models and the prediction MSE was computed. Figure 4.13 shows that the MSEs of the two ANNs (hydraulic head and risk ANNs) grow as the GA moves into later generations. The figure indicates that the GAs coupled with statically trained ANNs may not perform well due to the degrading ANN performance. Figure 4.14 compares the fitness values of a standard GA coupled with the statically trained ANN vs. the true PDE evaluated fitness for the

hypothetical case. The dashed line in the plot shows the true GA's fitness value at each generation (with no ANN), and the solid line shows the ANN+GA's estimated fitness values. The square dot shows the true fitness function value of the converged ANN+GA solution, found by evaluating that solution with the simulation models. Figure 4.14 shows that the true converged solution from the ANN+GA (evaluated with the simulation models) was much worse than what the ANNs predicted. In fact, the final population was within an infeasible region because the ANNs underestimated the risks. The ANNs predicted that the solutions were feasible when in fact penalties should have been incurred for violating the risk constraints. These false optimal solutions had slightly lower fitness than the true solutions, which in turn misled the GA to the wrong optimal solution. These results show the need for an adaptive retraining approach.

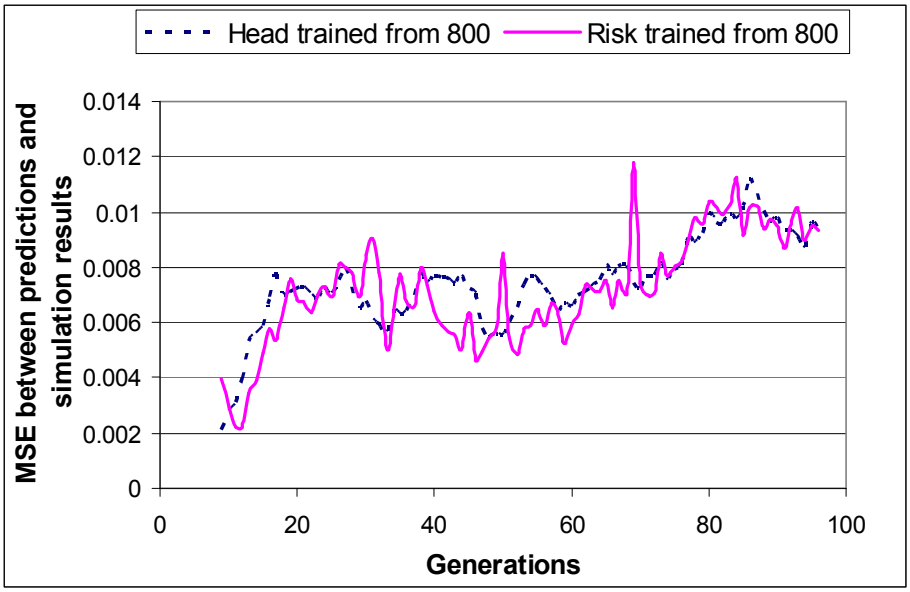

Figure 4.13. Testing MSEs for ANN+GA after training in 8th generation

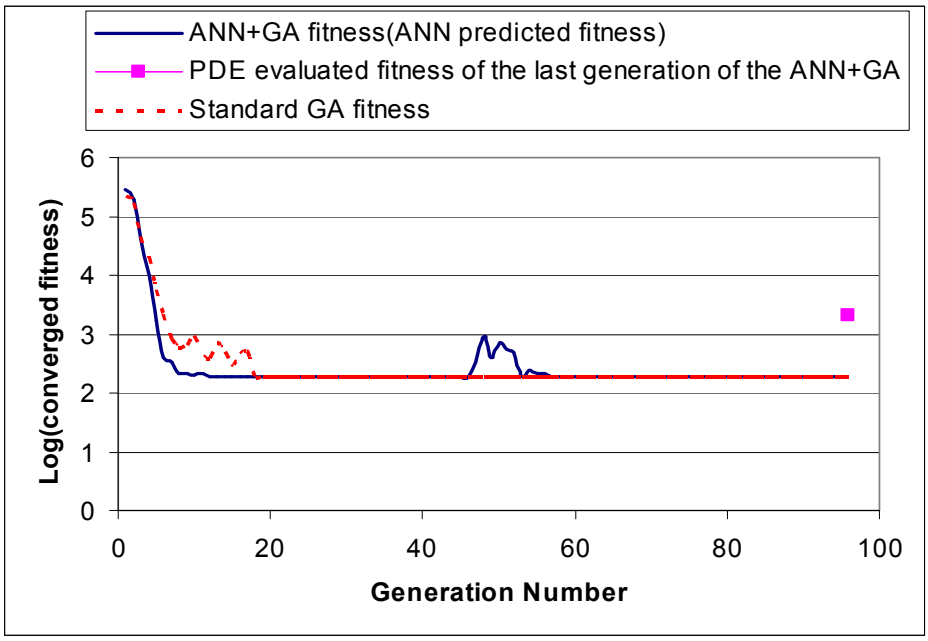

Figure 4.14. False convergence of a typical standard GA with statically trained ANN solving the hypothetical case.

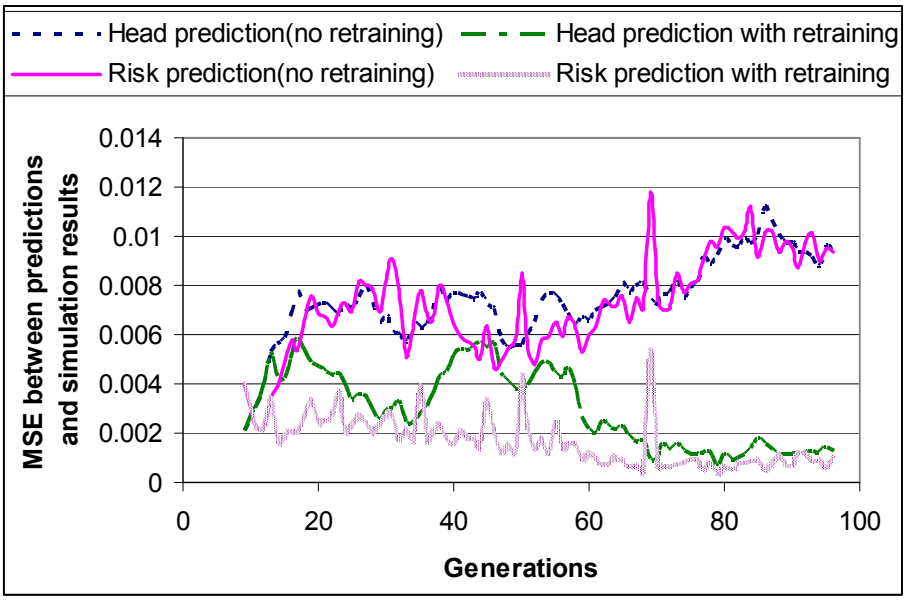

Figure 4.15. Testing MSEs of statically trained and retrained ANNs for the hypothetical case.

In comparison, Figure 4.15 compares the testing MSEs of adaptively retrained ANNs with the statically trained ANNs for both head and risk predictions. As indicated in the figure, the adaptively retrained ANNs performed much better in later generations.

The testing MSEs of the ANNs were consistently decreased by the retraining process and more accurate predictions were obtained.

AMGA Parameter Sensitivity Analysis Using the Hypothetical Test Case

AMGA has four parameters, defined in sections 4.3.2 and 4.3.3 as noted: the initial sampling rate (S_0, Equation 4.2-4.5), the sampling strategy, the initial training generations (G_0), and the retraining set management strategy. A sensitivity analysis was performed on the hypothetical test case to identify the best parameter settings, which were then applied to the Umatilla test case. During the analysis process, the total number of parameter combinations was reduced by adjusting one parameter and fixing the others at each run. For each parameter combination, AMGA was tested with ten random initial populations and the overall results were compared. Details on the findings are given below.

Initial Sampling Rate (S_0). The initial sampling rate determines the overall sampling level of AMGA and the retraining frequency of the ANNs. In our implementation, AMGA automatically adjusts its sampling rate according to the degree of population homogeneity, and the initial sampling rate, S_0, as shown in Equation 4.4. We tested AMGA's performance for values of S_0 equal to 0.05, 0.1, 0.2 and 0.3 (i.e., 5, 10, 20, and 30% of the population are initially sampled). Figure 4.16 shows AMGA's results for different values of S_0. The x axis is the run number, where each run number corresponds to a different random seed (initial population). The y axis is the converged objective function value across 10 initial populations.

For values of S_0 greater than 0.1, all AMGA runs converged to near global minima. When S_0 was decreased to 0.05, three of the ten runs converged to local minima. It is believed that this occurred because when the sampling rate is too low, the GA has insufficient true fitness values to ensure effective global searching at early generations. Low sampling rates also cause longer retraining intervals (assuming a fixed retraining pool size), thus the ANNs' initial coarse prediction was maintained for more generations.

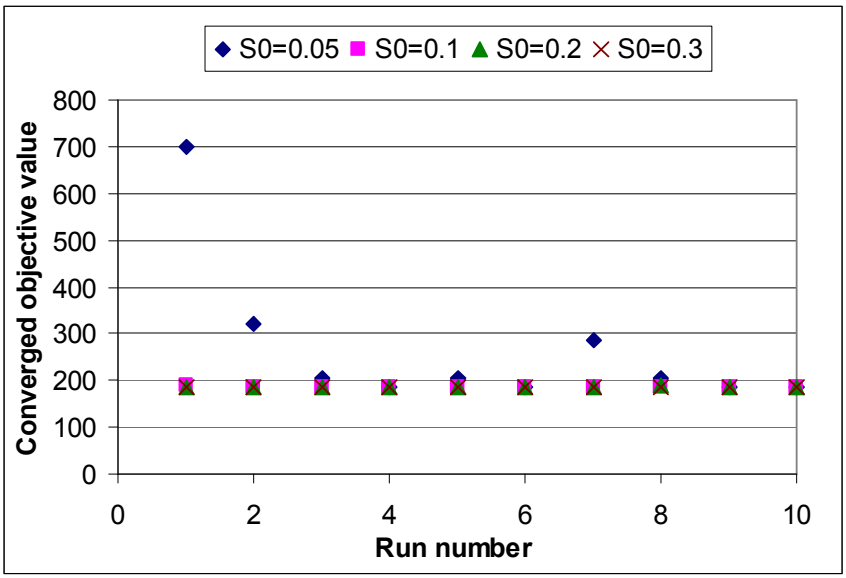

Figure 4.16. Optimal solution to the hypothetical case, found with AMGA for different initial sampling rates.

Figure 4.17 shows the average number of simulation model calls for the four sampling levels, with the error bars showing the maximum and minimum number of calls. As S_0 increases, the required overhead increases at a much lower rate. For example, when S_0 was increased 6-fold from 0.05 to 0.3, the simulation model calls increased only 2-fold. This is because the higher sampling rate at early generations caused quicker convergence, which in turn decreased *SSD* and the sampling rate more rapidly (see Equation 4.4).

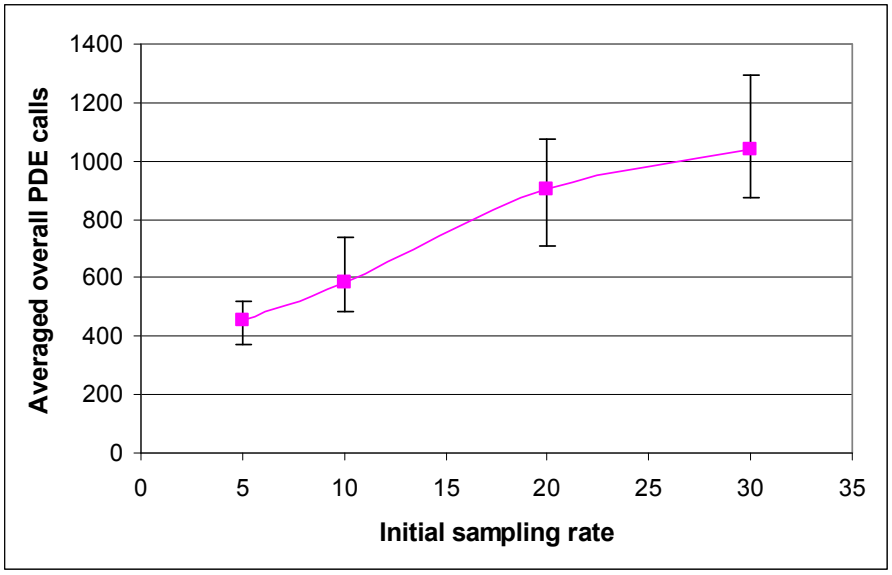

Figure 4.17. Number of PDE calls to solve the hypothetical case for different initial sampling rates, averaged across 10 random seeds. The error bars show the maximum and minimum number of PDE calls.

Sampling Selection Strategy. The four sampling selection strategies described previously (random sampling, best sampling, tournament sampling and tournament+best sampling) were tested with two sampling levels (S0=0.1 and S0=0.2). Figure 4.18 shows the results. The sampling selection strategies of best sampling and tournament+best sampling performed better at both sampling levels. Although the tournament+best sampling failed for the second random seed when S0=0.1, further random seed tests showed that the performance of this strategy was not very different from the best sampling strategy.

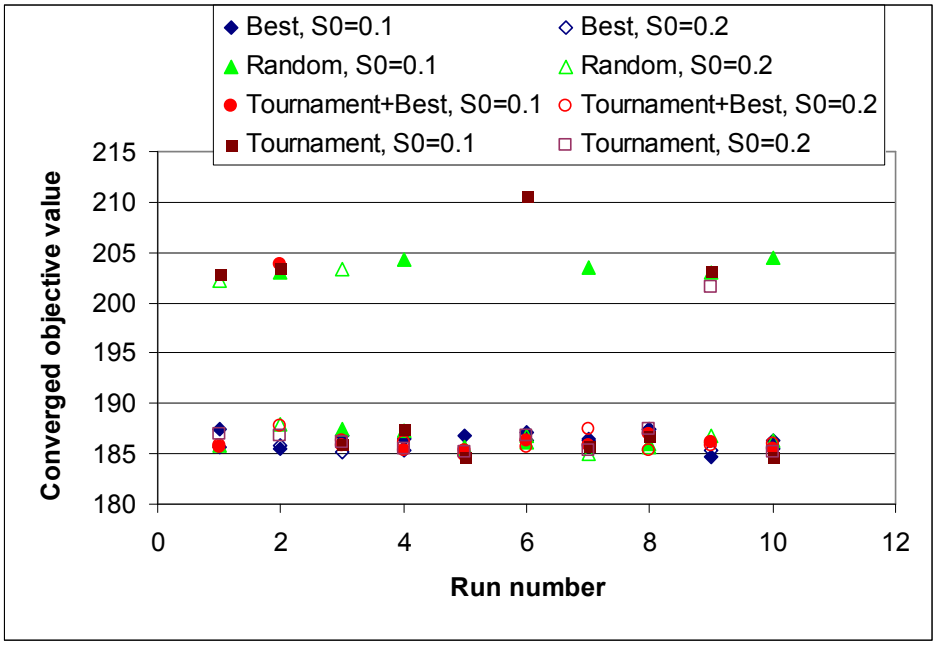

Figure 4.18. Optimal solution to the hypothetical case found with different sampling selection strategies and initial sampling rates.

Figure 4.18 also shows that the performance of tournament sampling dropped when S_0 was decreased to 0.1. This occurred because the global minimum existed on the feasible-infeasible constraint boundary. As AMGA approached convergence, the population was clustered at the feasible-infeasible border. Although the ANNs' predictions were quite accurate, they were not error-free and sometimes infeasible solutions were evaluated as feasible. When the sampling rate was low, these solutions were not corrected by numerical model evaluations. If instead the best sampling strategy was used at later generations (e.g. tournament+best sampling), false solutions with lowest fitness were always corrected and false convergence problems were avoided.

Initial Training Generations (G_0). The number of initial training generations, G_0, determines the initial training set size because the first ANN is trained on the solutions generated after G_0 generations. The training set size is G_0*N, where N is the population

size. We tested AMGA with values of G_0 equal to 3, 5, and 8, with corresponding initial training set sizes of 300, 500, and 800 fitness evaluations. Figure 4.19 shows that AMGA's performance is insensitive to the size of the initial training set within this range, since AMGA converged to near global minima for all three values of G_0. Therefore a coarse fitness estimation at early stages is sufficient for the GA to detect promising regions of the solution space. Smaller values of G_0 are preferred because fewer numerical model evaluations are required in the preprocessing stage. G_0 was therefore set to 3 for all other test runs.

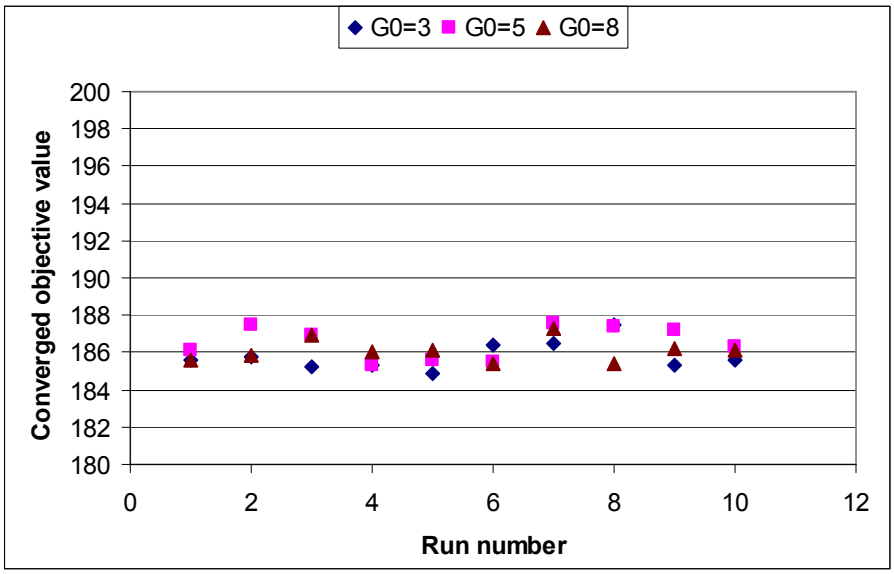

Figure 4.19. Optimal solution to the hypothetical case found with AMGA for different numbers of initial training generations.

Figure 4.20 shows the average number of PDE calls for the three values of G_0, with the error bars showing maximum and minimum numbers. When G_0 was increased from three to eight, 500 additional simulation model runs were required to prepare the

initial training sets. The overall simulation model calls, however, increased only about 300 times, due to faster reduction in *SSD* at later generations (equation 4.3).

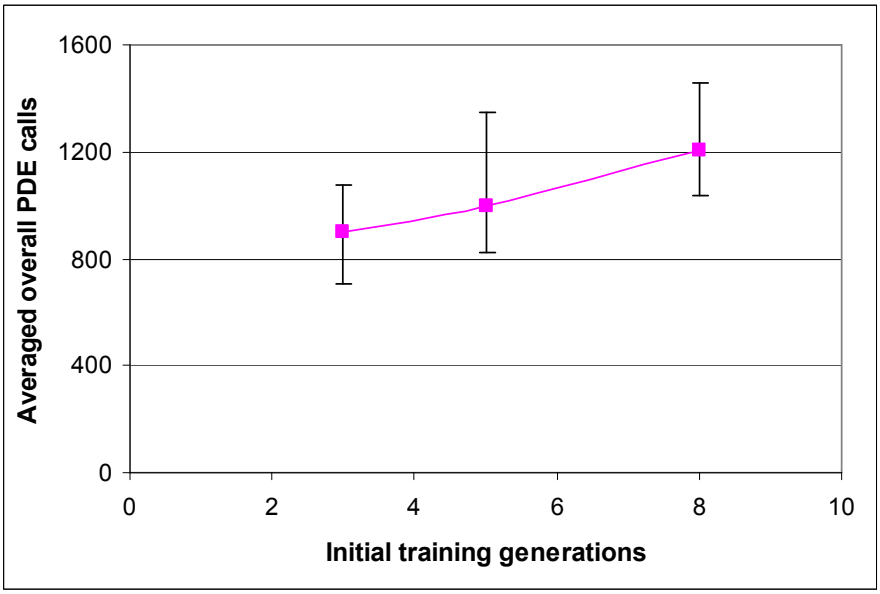

Figure 4.20. Number of PDE calls to solve the hypothetical case for different numbers of initial training generations, averaged across 10 random seeds. The error bars show the maximum and minimum number of PDE calls.

Retraining Set Management Strategy. Two strategies discussed previously, the fixed set approach and the growing set approach, were tested. Figure 4.21 shows AMGA's results with the growing retraining set versus the fixed retraining set approach. The results show that the fixed retraining set was more likely to converge to local minima than the growing set approach, confirming that too much local sampling can make the ANNs lose global approximation. When the training set size was fixed, early globally sampled points were replaced by new locally sampled points. The ANNs retrained by this training set approximated the local region accurately, but could not adequately predict the fitness values of solutions outside that region.

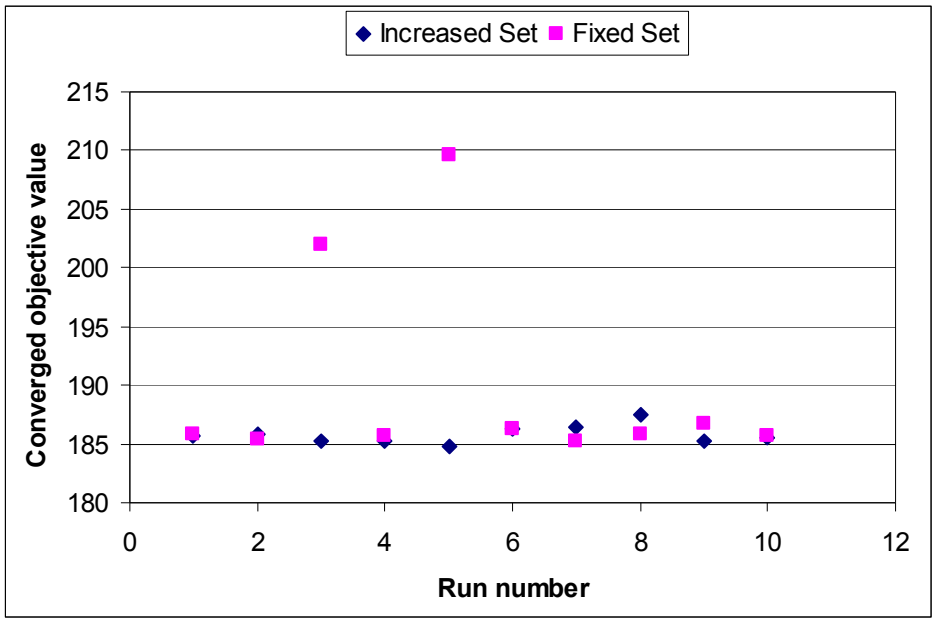

Figure 4.21. Optimal solution to the hypothetical case found with AMGA for different retraining set approaches.

GA and AMGA Comparison.

Once the best sampling and retraining set management strategies were found (S_0 set to 0.2, and G_0 set to 3), the hypothetical case was solved to compare AMGA's performance to a simple GA and a GA with caching (GA+CACHE). Ten initial random populations were tested for each algorithm. All three algorithms converged to a variety of local optimal solutions with total costs ranging from $185,600 to $187,000, and all constraints were satisfied. AMGA's solutions had somewhat lower objective function values than the GA for most seeds. Two pumping wells were recommended for installation in slightly varying locations for all 10 random seeds. As shown in Figure 4.22, the region in which the two pumping wells were recommended for installation is in the plume area, slightly shifted to the downgradient direction.

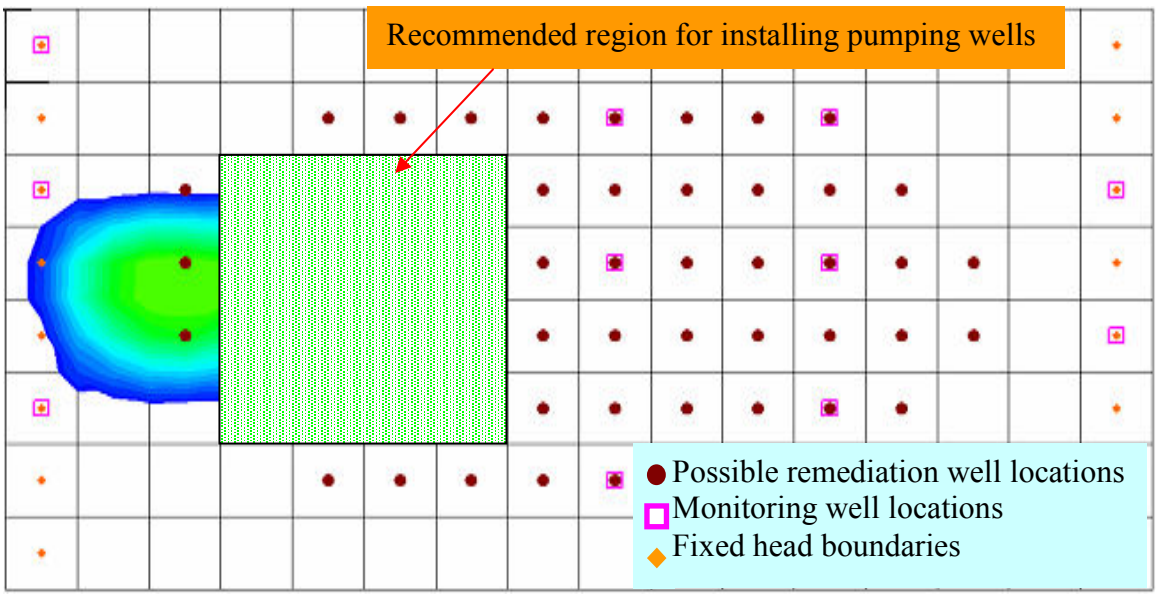

Figure 4.22. AMGA solution overview of the hypothetical case

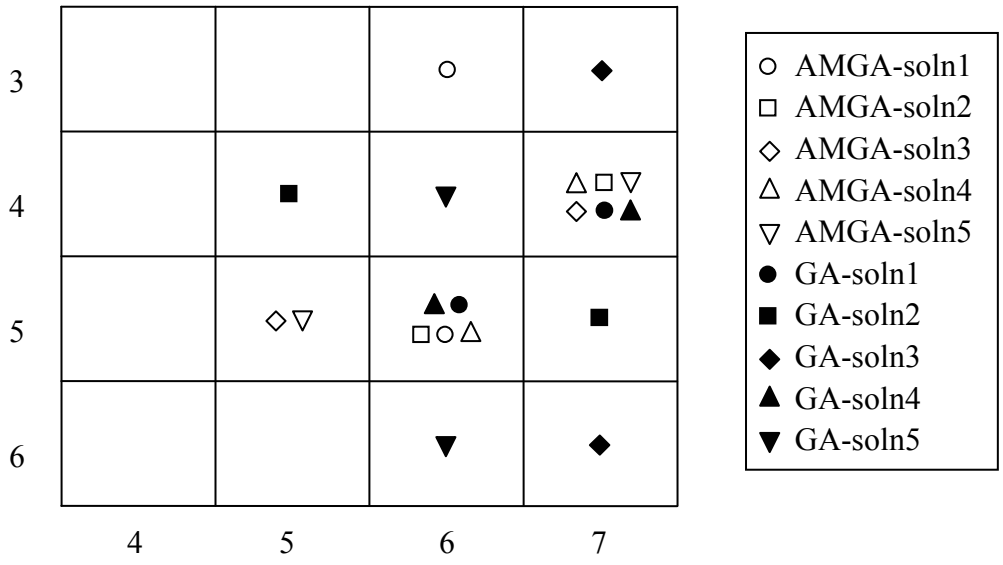

Figure 4.23. Well locations of the first five AMGA and the GA solutions for the hypothetical case study. The grid corresponds to the highlighted area in Figure 4.22.

Figure 4.23 and Table 4.3 give detailed well locations and pumping rates for the solutions found in the first five AMGA and GA runs. The other five runs converged to

similar solutions and are not shown here for brevity. Figure 4.23 shows that grid cells (6,5) and (7,4) are the two most likely cells to install the pumping wells. There are many other cell combinations nearby that can produce almost the same objective function value as long as the wells are installed within the critical region so that the contaminants can be effectively intercepted. Table 1 shows that the pumping rates of the installed pumping wells vary from 140 to 240 m3/d. However, the total pumping rates vary in a much smaller range (370 to 420 m3/d). Therefore the objective function values are somewhat less sensitive to the variation of the pumping rates at individual wells but more sensitive to the total pumping capacity. Note that GA+CACHE found the same solutions as the GA and therefore its solutions are not listed in the table.

Table 4.3. The first five solutions of AMGA and the GA for the hypothetical case.

Wells	Locations (col,row)	Pumping rates of AMGA solutions (m^3/day)					Pumping rates of the GA solutions (m^3/day)				
		Soln1	Soln2	Soln3	Soln4	Soln5	Soln1	Soln2	Soln3	Soln4	Soln5
Well-1	(5,5)			192		142					
Well-2	(5,4)							161			
Well-3	(6,6)										193
Well-4	(6,5)	224	192		176		164			224	
Well-5	(6,4)										226
Well-6	(6,3)	168									
Well-7	(7,6)								224		
Well-8	(7,5)							228			
Well-9	(7,4)		192	192	198	236	227			192	
Well-10	(7,3)								192		
Total extraction		392	384	384	374	378	391	389	416	416	419
Total cost ($K)		185.6	185.5	185.2	185.3	184.9	186.1	185.2	187.1	187.0	187.0

For the ten initial populations, the AMGA runs required 707 to 1,088 simulation model evaluations, with an average of 900 evaluations. Figure 4.24 shows the

performance comparison of the fifth random seed for the GA, GA+Cache and AMGA in terms of number of simulation model evaluations. While all three methods found similar solutions, the GA evaluated the numerical models approximately 9,600 times, the GA+cache required about half that many evaluations, and AMGA needed only 808 evaluations. Compared to the simple GA, AMGA saved more than 90 percent of the simulation model runs required to locate the near optimal solutions. Compared to the GA with caching approach, AMGA saved more than 80 percent of simulation model runs.

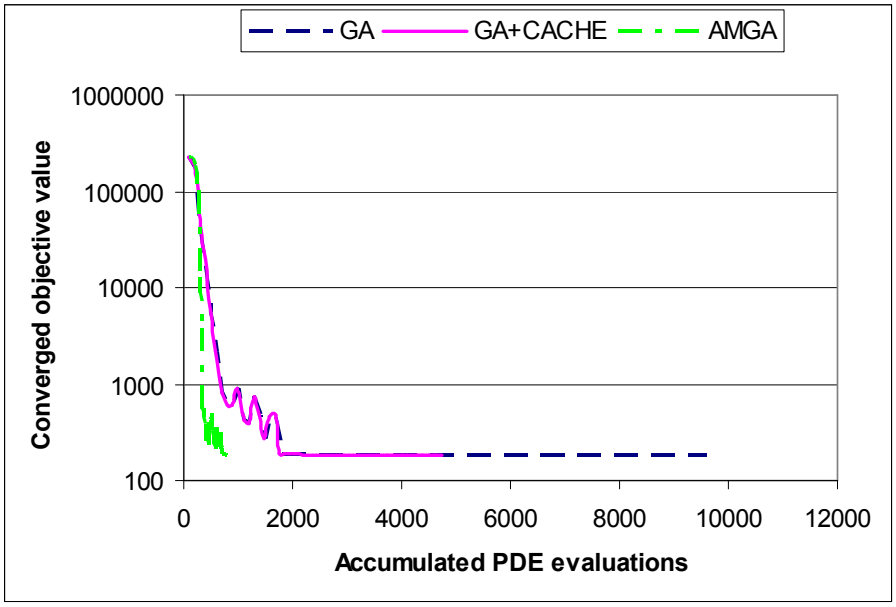

Figure 4.24. Number of fitness evaluations required for GA, GA+Cache and AMGA to solve the hypothetical case for one random seed.

Figure 4.25 shows how the population's fitness evaluations were performed in each generation of a typical AMGA run. At the preprocessing stage, which consists of the first three generations, the numerical models performed most of the fitness evaluations. After three generations, the ANNs were trained and they began to dominate the fitness

evaluations. During the second half of the run, the populations were clustered within local regions, and the cache became more active, bypassing most fitness evaluations.

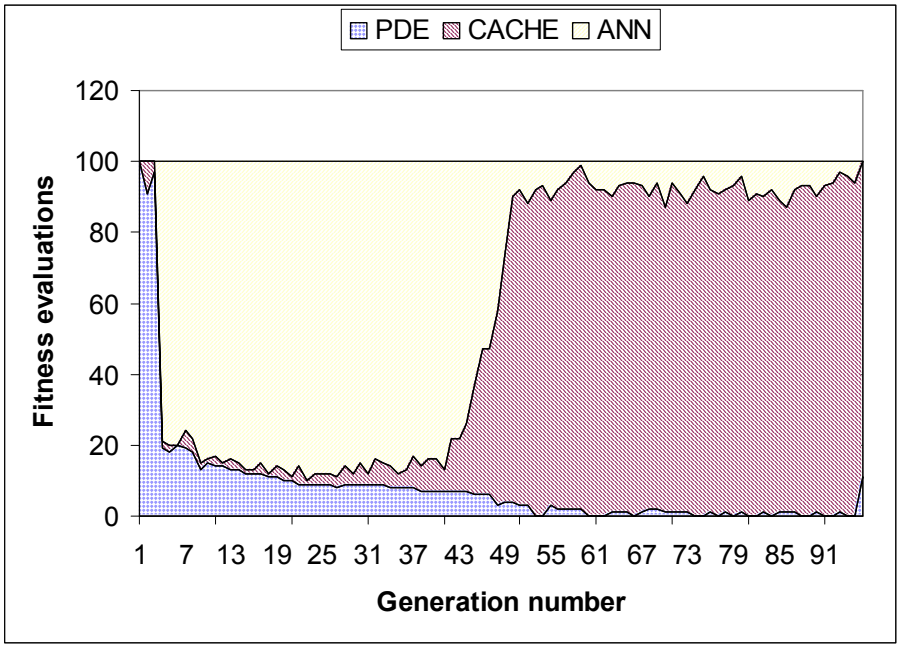

Figure 4.25. Fitness evaluations for a sample run of the hypothetical case with AMGA.

4.5.2 Applying AMGA to the Umatilla Case

In this section, the findings of the previous section are tested on the Umatilla case, which is significantly more complex than the hypothetical case. First, the hydraulic conductivities vary within a much larger range across the site, giving the flow and transport model a non-smooth response surface to pumping in different locations, with more local minima. Second, the case has 14 candidate wells or infiltration basins (seven new and seven existing), giving the ANNs more complex structures. Third, the objective function is a highly non-linear function that depends heavily on the number of cleanup years. Finally, the simulation models are much more computationally intensive and would be too costly to test on every parameter combination as with the hypothetical case.

Here, only the best parameter combinations found in the hypothetical test case were applied to the Umatilla case. The initial sampling rate S_0 was 0.2, the initial training generations G_0 was 3. The retraining set grew as new points were sampled. The best sampling and tournament+best sampling strategies were used.

AMGA Convergence for Different Sampling Selection Strategies. Figure 4.26 shows the results applying the above parameters to the Umatilla case. Unlike the hypothetical case where both best sampling and tournament+best sampling strategy worked well, only tournament+best sampling worked well for all ten random seeds. This phenomenon was attributed to the non-smooth step objective function and the relatively small local region where the global minimum exists. The optimized solution for the Umatilla case needs four years to clean up the contamination. Minsker et al (2003) reported that hundreds of slightly different solutions exist that yield virtually the same objective function value. These solutions appear to cluster in a small region. The region is surrounded by five-year cleanup solutions and the objective function quickly steps up when moving outside of the global minimum region (note that the objective function value is proportional to the cleanup duration). At early generations, the GA was unlikely to randomly generate points in the small region containing the global minimum, thus early ANNs could not learn the functional relationship in the region. The trained ANNs then erroneously predicted all solutions needing five years to clean up and therefore overestimated the objective function values for the four-year solutions. When best sampling was applied at an early stage, this strategy preferred the regions with local

minima containing five-year solutions. The GA was then misguided into the wrong local region, the ANN adapted to this local region accordingly, and the GA never found the four-year solutions. If instead tournament sampling were used, this strategy had much wider sampling coverage while still exploiting the knowledge of ANNs. Hence the four-year solutions can still be chosen for simulation model evaluations. Once ANNs were updated based on these points, AMGA was led to the correct region where the global minima lie. Thus better performance was observed for the tournament+best sampling strategy.

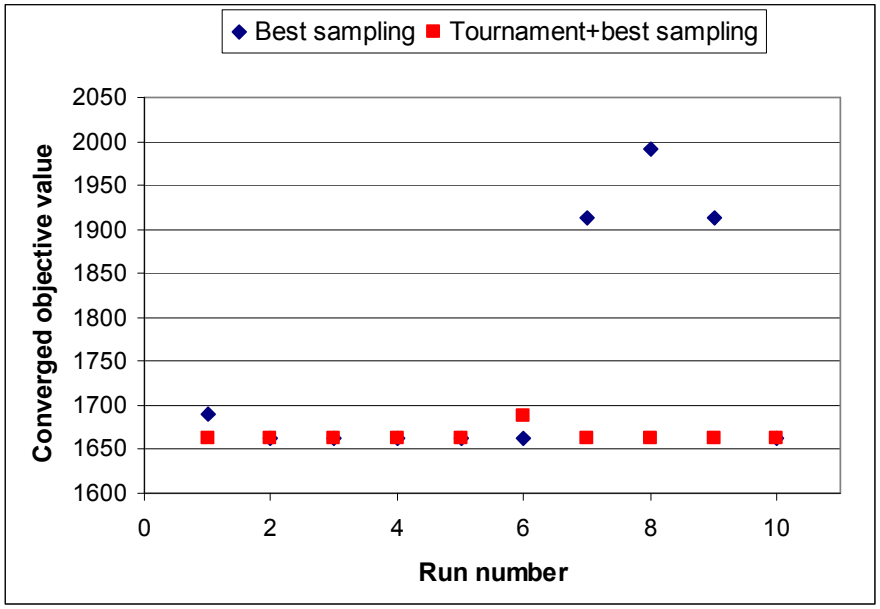

Figure 4.26. AMGA convergence for different sampling selection strategies

Using these settings, we solved the Umatilla case with both AMGA and the GA. The objective function value of the optimal solutions found by both algorithms was in the range of 1.662 to 1.664 million dollars over a span of 4 years, which was somewhat better than the solutions reported by the ESTCP team (1.664 million). The optimal

solutions found that two existing pumping wells in the TNT plume should be used, and two new wells should be installed within the TNT plume. The existing recharge basin located toward the north and the two extraction wells located within the RDX plume were not used. The two existing recharge basins located toward the south were instead used to flush the RDX plume toward the TNT plume so that eventually both plumes could be removed by the extraction wells. This solution is consistent with the ESTCP team's recommended cleanup strategy. Moreover, like the hypothetical test case, the problem has many local minima, because the two new wells can be installed in slightly different locations in the region shown in Figure 4.27 with similar effectiveness. The phenomenon that multiple well locations produce equally good solutions was also reported by Peralta (2002). Future runs may be required to test if dramatically different well configurations can yield similar low cost solutions.

Figure 4.28 shows the detailed locations of the new pumping wells for each solution. The pumping rates of the wells and recharge basins are listed in Table 4.4 for both AMGA and the GA runs. Again for brevity, only the first five solutions are shown here. The total pumping rate (1044 ~1147 gallons/min) is slightly smaller than the ESTCP optimal results (1170 gallons/min), which probably contributes to the slightly lower objective function values. As shown in Figure 4.28, the two new wells are most likely to be installed in cells (55, 48), (59, 48) and (57,49). The first two locations were reported by one optimization team in the ESTCP report (Minsker, 2003), confirming that both AMGA and the GA found good solutions.

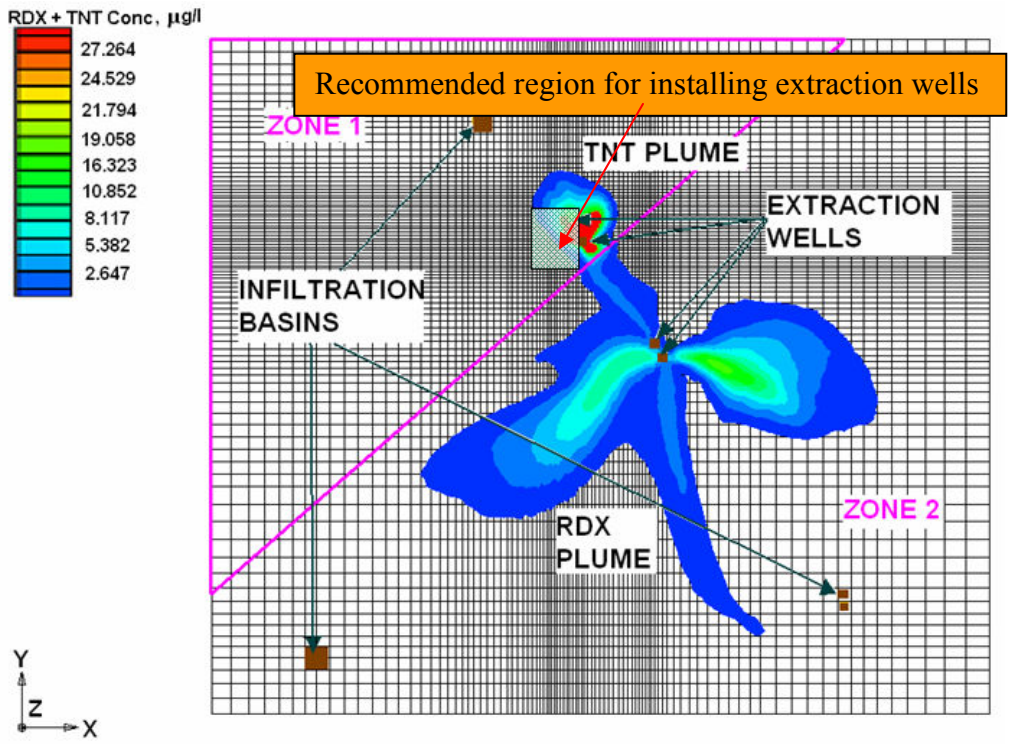

Figure 4.27. AMGA solution overview of the Umatilla case study.

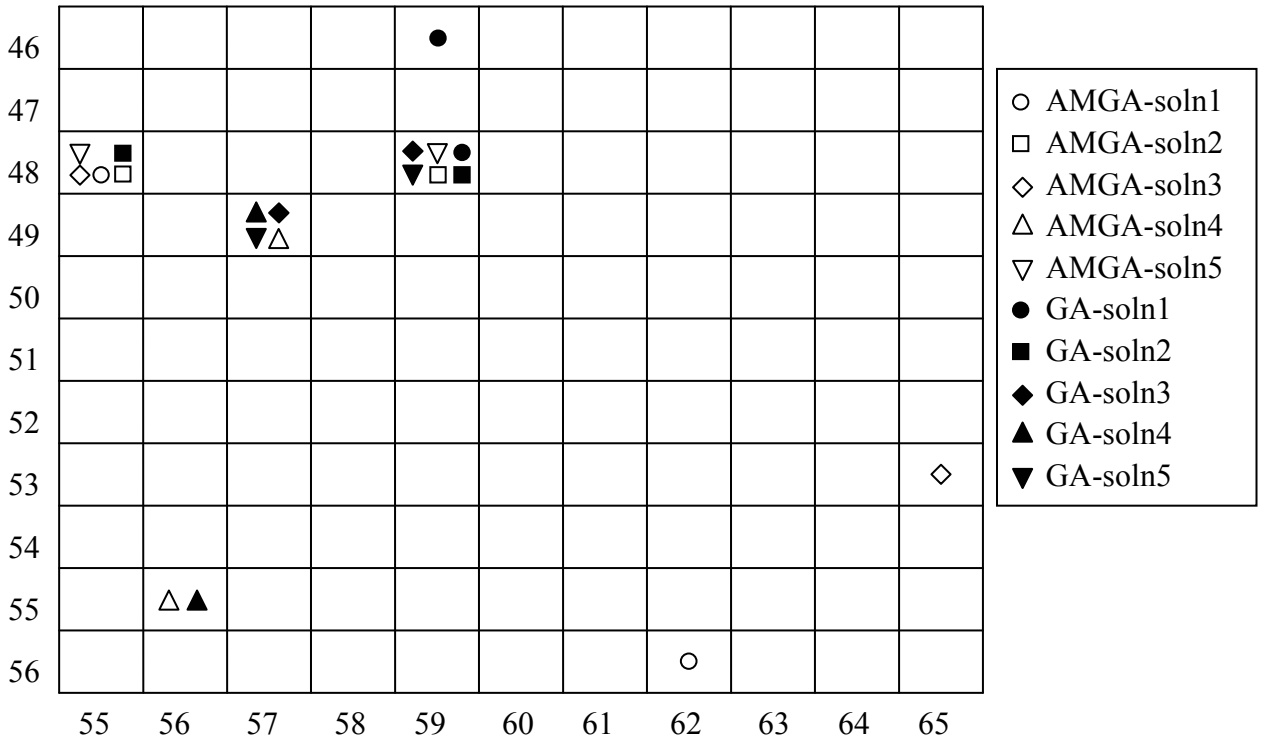

Figure 4.28. Well locations of the first five AMGA and GA solutions for the Umatilla case study. Grid corresponds to the highlighted area in Figure 4.27.

Table 4.4. The first five solutions of AMGA and the GA for the Umatilla case.

Wells	Locations (col,row)	Pumping rates of AMGA solutions (gallons/minute). Negative pumping rates imply extraction wells.					Pumping rates of the GA solutions (gallons/minute). Negative pumping rates imply extraction wells.				
		Soln1	Soln2	Soln3	Soln4	Soln5	Soln1	Soln2	Soln3	Soln4	Soln5
Well-1	(59,46)						-329				
Well-2	(55,48)	-303	-104	-354		-79.5		-79.5			
Well-3	(59,48)		-353			-342	-136	-353	-320		-321
Well-4	(57,49)				-358				-223	-335	-248
Well-5	(65,53)			-127							
Well-6	(56,55)				-193					-276	
Well-7	(62,56)	-137									
EW-1		-342	-331	-222	-251	-359	-351	-339	-249	-165	-246
EW-2											
EW-3		-358	-350	-341	-333	-351	-314	-359	-341	-360	-317
IF-1											
IF-2		380	382	348	378	377	376	377	378	379	377
IF-3		760	764	696	756	754	753	753	755	757	754
Total extraction		-1140	-1147	-1044	-1134	-1131	-1129	-1130	-1133	-1136	-1131
Total cost ($M)		1.6630	1.6627	1.6630	1.6629	1.6628	1.6628	1.6627	1.6628	1.6629	1.6628

Figure 4.29 shows the converged fitness value for the two methods versus the number of simulation model evaluations. For this case study, AMGA required about 1600 numerical model evaluations, which saved more than 85 percent of the numerical model evaluations required by the GA to find a similar optimal solution.

Finally, Figure 4.30 shows the proportion of simulation model evaluations, ANN evaluations, and calls to the cache in each generation. The cache saved almost 1/3 of the numerical model calls in later generations when the populations became more homogeneous. However, the cache was not as effective as it was for the hypothetical test case. This is likely because the Umatilla case had a much longer chromosome (131 bits

for the Umatilla case, but only 45 bits long in the hypothetical case) and hence a much larger number of potential solutions were possible.

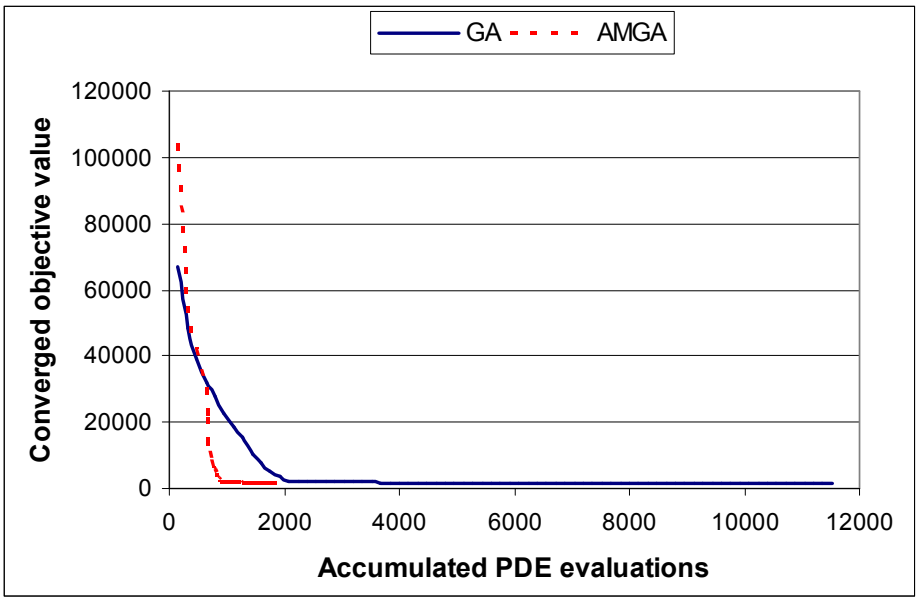

Figure 4.29. Number of fitness evaluations required for AMGA and GA to solve the Umatilla case

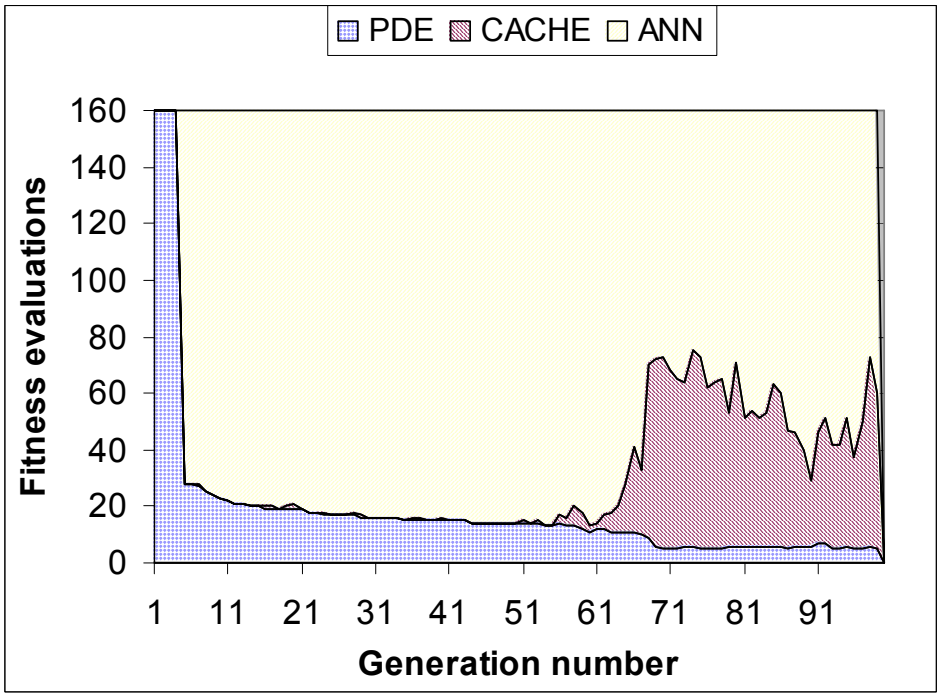

Figure 4.30. Fitness evaluations of the Umatilla case with AMGA for one sample run.

4.6 Conclusions

This chapter demonstrates how a statically trained ANN can introduce errors into the dynamic environment of the GA where the population moves from global search to local search near the optimal solution. An adaptive meta-model genetic algorithm called AMGA was proposed, which adaptively retrains the ANN-based meta-models. The adaptive retraining approach presented here allows the ANNs to gradually move from global to local approximation by injecting additional local sampling solutions into the training set. This corrects the ANN predictions and the GA's convergence. The smoothed scaled standard deviation parameter provides a convenient mechanism for computing the homogeneity of the GA's population and adjusting the sampling rate accordingly.

A caching technique implemented with AVL tree was also introduced, which efficiently retrieves previously evaluated solutions from an memory structure. This approach ensures that the most accurate fitness information is used without significant computational effort, which is most beneficial at later stages of the GA's run. The results showed that combining the caching technique with the adaptive ANNs can bypass most simulation model evaluations without affecting the GA's global searching ability.

The algorithm was tested on a hypothetical remediation test case with multiple random initial populations to identify the best AMGA settings. These settings also performed well on the more complex Umatilla case, using the tournament+best sampling selection strategy. Although both case studies have many local minima with respect to the pumping well locations and the variation of pumping rates at individual pumping wells,

the comparison of AMGA and the GA results shows that the two algorithms found solutions with virtually the same effectiveness and similar configurations. These findings are based on the assumption that the underlying flow and transport models were carefully calibrated and their predictions approximated the true contaminant migration very well. Should the simulation models be recalibrated or modified, the runs must be redone to identify new solutions. The methodology discussed in the chapter, however, still applies.

Overall, AMGA was able to save 85-90% of the time-consuming fitness evaluations required to solve the case studies while still achieving accurate solutions. These results indicate that AMGA should be tested on additional water resources case studies to verify these findings.

CHAPTER 5

TRUST REGION-BASED META-MODELS FOR WATER RESOURCES

In Chapter 4, artificial neural networks (ANNs) were used as efficient meta-models to approximate and replace groundwater flow and transport simulation models in AMGA. The predicting ability of the meta-models is important for achieving accurate solutions as AMGA relies on their predictions to estimate the qualities of trial designs. In this chapter, a technique that can improve the prediction ability of conventional meta-models is explored. The motivations of the technique are presented in the following section.

5.1 Introduction

Developing realistic models is of significant importance in groundwater hydrology, water-shed management, water quality prediction, and reservoir operations. During the past decades, our understanding of the underlying hydrological mechanisms has considerably improved, and a number of simulation models have been developed using equations of mass, momentum, and energy to mimic the physical processes in nature. However, modeling large-scale and complex hydrological problems is still a challenge due to our incomplete knowledge of the interrelated physical processes and the difficulty of charactering and quantifying the physical and chemical parameters. Moreover, realistically modeling complex hydrological processes usually entails the solution of computationally extensive partial differential equations. This time

requirement could be a barrier when simulation models are incorporated into water resources management, as the simulation models are repeatedly evaluated to assess the candidate design solutions. One approach to tackling this computational issue is the dynamic genetic algorithm (GA) framework proposed in Chapter 3. The framework combines GAs with adaptively updated artificial neural-networks so that numerical model evaluations can be substantially reduced.

Using ANNs to approximate simulation models is a typical application of data-driven models. In addition to providing computationally inexpensive response meta-models, data-driven models are also used to provide realistic predictions in cases where conceptual models fail to fully capture the underlying physical processes. The data-driven models are in the category of black-box models, which learn the functional relationship mapping inputs to outputs from a given training set. Most of the data-driven models use a global training algorithm to learn the functional relationship from a given training data set. During the training phase, a global error function measuring the models' overall prediction performance is usually used to control the selection of model parameters. In some work, a regulation term is included in the global error function by penalizing the complexity of model parameters in order to avoid over-fitting complex model structures (Hastie et al, 2001). Such an error function is a necessary property of the training algorithm so that it can tolerate noise existing in the training sets by smoothing off local details and low frequency values. However, accurate approximation of local details or low frequency values may sometimes be desired. For instance, when meta-models are

used in GA-based optimization models, the accuracy of the meta-models in the local region where the global minimum is located directly affects the accuracy of the GA's solution. Therefore, training the meta-models only by controlling the global error functions is not always necessarily the best choice.

A promising approach to handling this issue is the divide-and-conquer method, which tackles complex problems by dividing them into smaller and easier-to-solve sub-problems and then combining the sub-solutions to yield the solutions to the original problems. The method has the advantage that predictions in local regions may be improved by local models devoted only to those regions. The divide-and-conquer method has been used for classification (decision tree) and function approximation (Jacobs and Jordan, 1993; Jordan and Jacobs, 1994; and Zhang and Govindaraju 2000) as discussed in Chapter 2. Another concept for managing local models is the trust region method proposed in the optimization field. The trust region method manages approximation models with different fidelities so that the most appropriate model is chosen for prediction according to the trust regions of the prospective models. During the optimization, the trust region is adaptively updated to avoid over-trust or under-trust of the approximation models (Alexandrov et al, 1998). This method can be combined with the divide-and-conquer concept so that local models and a global model can be properly managed to contribute to function approximations.

This chapter builds on these concepts by proposing a trust region based meta-model framework. Instead of creating a single global meta-model, the method

hierarchically clusters the data set into local subsets using a cluster algorithm. A global meta-model and a series of local meta-models are then trained at each hierarchical level to make predictions. The meta-models trained on local regions are trusted only in these regions, and trust region testing is used to select appropriate meta-models for prediction. The method has the advantage that the local regions approximated by more accurate local meta-models can be more finely modeled and better solutions may be obtained. This approach is first demonstrated with the dynamic GA optimization framework presented in Chapter 4 for the Umatilla remediation case study, comparing performance with both ANNs and SVMs as the prediction tools (Section 5.3.1). Next, the method is tested on a general nitrate prediction problem using a published case study (Sections 5.3.2). The results of the two case studies are presented in Section 5.4. Finally, Section 5.5 concludes the chapter.

5.2 Methodology

The trust region based meta-model framework is motivated from the classical trust region optimization algorithm. Here the background on the trust region optimization algorithm is briefly introduced in Section 5.2.1. Then the idea is extended to construct trust region-based hierarchical meta-models for general water resources management in Section 5.2.2, as well as within the AMGA framework presented in Chapter 4 (Section 5.2.3).

5.2.1 Trust Region Optimization Algorithm.

The classical trust region approach is primarily used within non-linear optimization algorithms to improve global convergence. In most non-linear optimization methods, a local quadratic approximation model is established in the vicinity of the current solution. A quadratic model is then used to identify the best search direction and the size of the search step to be taken in that direction. The trust region algorithm enforces a trust region in which the search step sizes and search directions may be adjusted to take best advantage of the approximation model's prediction without moving beyond the region within which the quadratic approximation is valid. In practice, the trust region is updated from step to step: if the approximation model gave accurate predictions of the improvement in the objective function, then the region is enlarged. If the approximation model gave poor predictions, then the region is decreased to improve the fidelity of the quadratic approximation.

The classical trust region algorithm was designed to solve optimization problems that can be approximated by quadratic functions. Alexandrov et al (1998) proposed a trust region framework for managing more general approximation models. In this framework, a series of approximation models with different fidelities, or established for different sub-problems, can co-exist. In addition to adjusting the trust region size, the framework may switch the approximation models so that a model with lower computational cost but acceptable fidelity is chosen for prediction.

5.2.2 Trust Region-based Hierarchical Meta-model

The concept of combining models of different fidelities to provide better prediction can be extended for meta-model construction. Generally meta-models have no extrapolation ability, and subsequent predictions must be confined to points contained within each meta-model's training region. Here we define a trust region of a trained meta-model as the region covered by its training set. The region must cover enough training points so that meta-models can be trained with good prediction accuracy. The size of a region can be represented by the radius of the smallest hyperellipsoid containing the training points. If there are enough training points, a number of meta-models that each covers a different trust region can be assembled into a hierarchical relationship. At the top level, a global model is trained by the global training set, which is trusted for any prediction but may have low accuracy in certain regions. The local models at lower levels have limited trust regions but presumably higher fidelity because the local models may be trained with smoother response surfaces and consequently lower prediction errors. When new points are to be predicted, a trust region test can be used to choose the trusted approximation models, which predict the outputs. In the case where multiple models are selected, their outputs can be blended to generate the final predictions. Details of each step will be addressed in the following subsections.

Model Construction. The trust region-based hierarchical model is constructed from the top level down to the bottom level. At each level, the local clusters in the training set must be identified so that the current training region can be divided into local

regions. Figure 5.1 illustrates the concept using a hypothetical training set. At the top level, a single global model *A* is trained on all sampled training points. In this training set, we assume two local clusters exist, within which two sub-models are locally trained at the second level with training points from regions *B1* and *B2*. The sub-models presumably have better fidelity in these regions. At the third level, two more specific models (*C1* and *C2*) provide the highest accuracy in those smallest regions. Although more detailed local models can be continuously generated, the number of levels of the hierarchical model and the number of local regions are often limited by the available training points and their distributions. In practice, there is often a tradeoff – local models are more capable of capturing smoother local surfaces but scarce training points in local regions increase the risk of over-fitting.

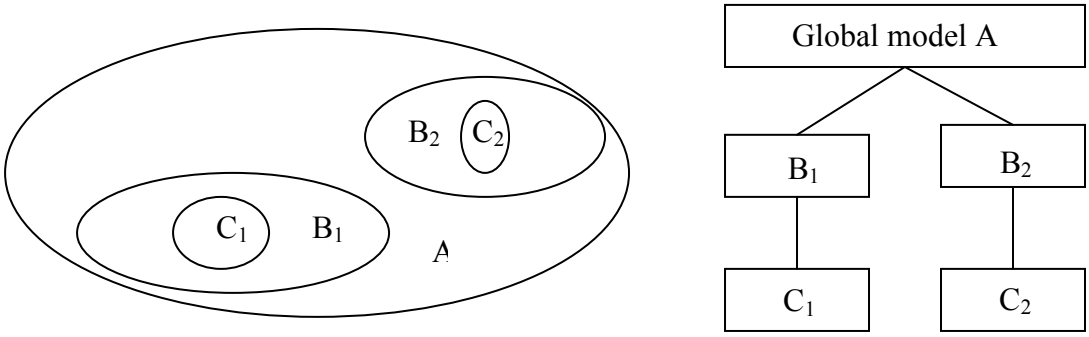

Figure 5.1. Trust region based hierarchical model

Model Training and Prediction. Although the sub-models of the trust region approach in Figure 5.1 are organized in a hierarchical relationship, the training of each sub-model is independent and there is no limitation on what training algorithm can be used. The sub-models must be combined, however, when a prediction is to be produced.

The easiest approach to combining the models is the winner-takes-all-strategy: if multiple sub-models are trusted, the most detailed local model is the one that produces the prediction. The pitfall of this method is that dividing the data space into sub-spaces generally increases the prediction variance (a prediction error can always be decomposed as the superposition of prediction bias and prediction variance [Hastie, 2001]), although it has favorable consequences for the prediction bias of the local models (Jordan and Jacobs, 1994). This effect is more serious at the local boundaries if there are no training points to control the function shape at or near the boundaries when the local models are trained. A better alternative is to mix the outputs of all trusted sub-models using a soft blending function. Note that the soft blending method can ameliorate the variance increasing effect of the hierarchical model as the higher level models (more general models) can contribute to damping the variance caused by the lower level models (more detailed models).

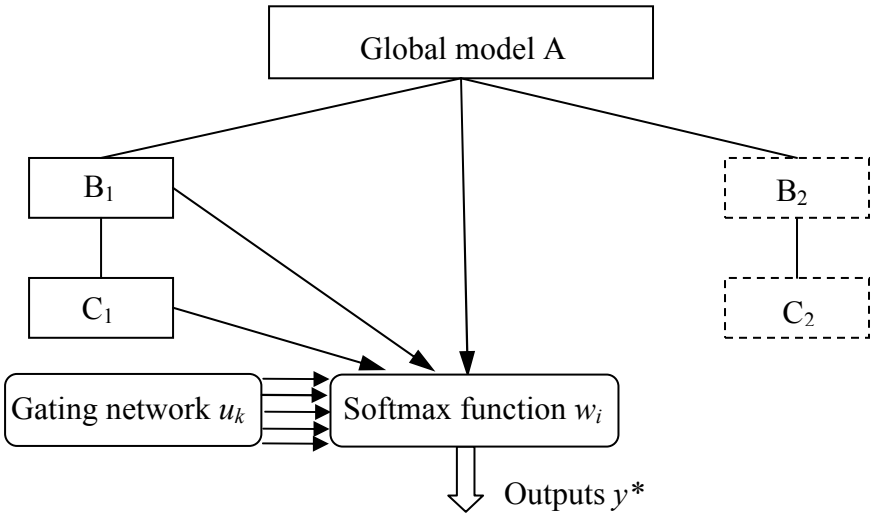

Figure 5.2. Gating network in the trust region-based hierarchical model

The soft blending method requires computing the contribution from each trusted sub-model. Jordan and Jacobs (1994) proposed a gating neural network to compute blending contributions and Zhang and Govindaraju (2000) used the technique to predict watershed runoffs. This technique is modified in this work to incorporate a trust region test in the hierarchical model structure. Figure 5.2 shows the relationship of the gating network to the sub-models. The solid lines show the hierarchical model and the solid arrows show the outputs from the sub-models. An example of trusted models and models that are not trusted (dashed box) are also shown. The gating network is a simple one layer neural network that receives the input features and outputs the contributions (u_k) of the sub-models. These contributions represent the likelihood of trusting the sub-models and they are then blended by the so-called softmax function (Jordan and Jacobs, 1994) to compute the weight coefficients (w_i), and the final output is the weighted average of the trusted sub-models. The softmax formulations are listed below,

$$y^* = \sum_{i}^{M} w_i y_i \qquad \text{- the weighted average of trusted sub-models} \qquad (5.1)$$

$$w_i = \frac{\exp(u_i)t_i}{\sum_{1}^{n} \exp(u_k)t_k} \qquad \text{- the softmax function to compute the weight coefficients} \qquad (5.2)$$

$$u_k = \left(a_{*,k}\right)^T x \qquad \text{- the predicted contribution for sub-model } k \qquad (5.3)$$

where x is the input vector; y^* is the final output prediction; y_i is the output from sub-model i, w_i is the weight coefficient of sub-model i; u_k is the contribution of sub-model k predicted by the gating network; M is the number of sub-models; t_i is the trust region

testing indicator, which is 1 if x is within the trust region of sub-model i and 0 otherwise; $a_{*,k}$ is the k^{th} column of the weight matrix a in the gating network.

The gating network is trained to update the weight matrix a after the sub-models are trained. A fast conjugate training algorithm to train the gating network is used in this research. The conjugate algorithm uses the Fletcher-Reeves formula to compute search directions, which is,

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} d_k \tag{5.4}$$

where d_k is the search direction in iteration k and g_k is the derivative (training gradient) of the training objective function to the weight matrix in the gating network. Details of conjugate gradient methods are provided by Polak (1971). To compute the derivative g_k in the training iterations, the formulas are derived as shown below using similar techniques presented by Jordan and Jacobs (1994),

The derivatives of equation (5.3) are,

$$\frac{\partial u_k}{\partial \mathbf{x}} = a_{*,k} \qquad \text{- the derivative of the input variable} \tag{5.5}$$

$$\frac{\partial u_k}{\partial a_{*,k}} = \mathbf{x} \qquad \text{- the derivative of the weight coefficients} \tag{5.6}$$

The derivatives of the softmax function (equation 5.2) are,

$$\begin{aligned} \frac{\partial w_i}{\partial u_i} &= w_i(1 - w_i) \\ \frac{\partial w_i}{\partial u_j} &= -w_i w_j \quad (i \neq j) \end{aligned} \tag{5.7}$$

The error is defined to be the difference between the observed value *d* and the predicted value *y*. The squared error loss function is then,

$$e = (d - y)$$
$$l = \sum_{i=1}^{N} (d - y)^T (d - y) \tag{5.8}$$

where *d* is the observed outputs, *e* is the error function, and *l* is the squared error loss function. Then the derivative of the loss function with respect to the weight matrix in the gating network is,

$$g_k = \frac{\partial l}{\partial a_{*,k}} = -2\sum_{p=1}^{N} \frac{\partial y}{\partial a_{*,k}}(d - y) = -2\sum_{p=1}^{N}\sum_{j=1}^{g} \frac{\partial w_j y_j}{\partial a_{*,k}}(d - y)$$
$$= -2\sum_{p=1}^{N}\sum_{j=1}^{g} \frac{\partial u_k}{\partial a_{*,k}} \frac{\partial w_j}{\partial u_k} y_j^T (d - y) = -2\sum_{p=1}^{N} x\left[-\sum_{j \neq k} w_j w_k y_j^T + y_k^T w_k (1 - w_k) \right](d - y) \tag{5.9}$$
$$= 2\sum_{p=1}^{N} x w_k \left[\sum_j w_j y_j^T - y_k^T \right](d - y) = 2\sum_{p=1}^{N} x w_k \left[y^* - y_k^T \right](d - y^*)$$

After the training is finished, equations (5.3), (5.2) and (5.1) can be used to blend the trusted sub-model predictions.

5.2.3 Trust Region-based Meta-model Within Adaptive GA

The AMGA framework presented in Chapter 4 is a general design that can incorporate other prediction models. In this section, a methodology for replacing the single ANN by the trust region based meta-model is proposed. An overview of the modified algorithm, the Trust Region based Adaptive Meta-model Genetic Algorithm (TRAMGA) is shown in Figure 5.3.

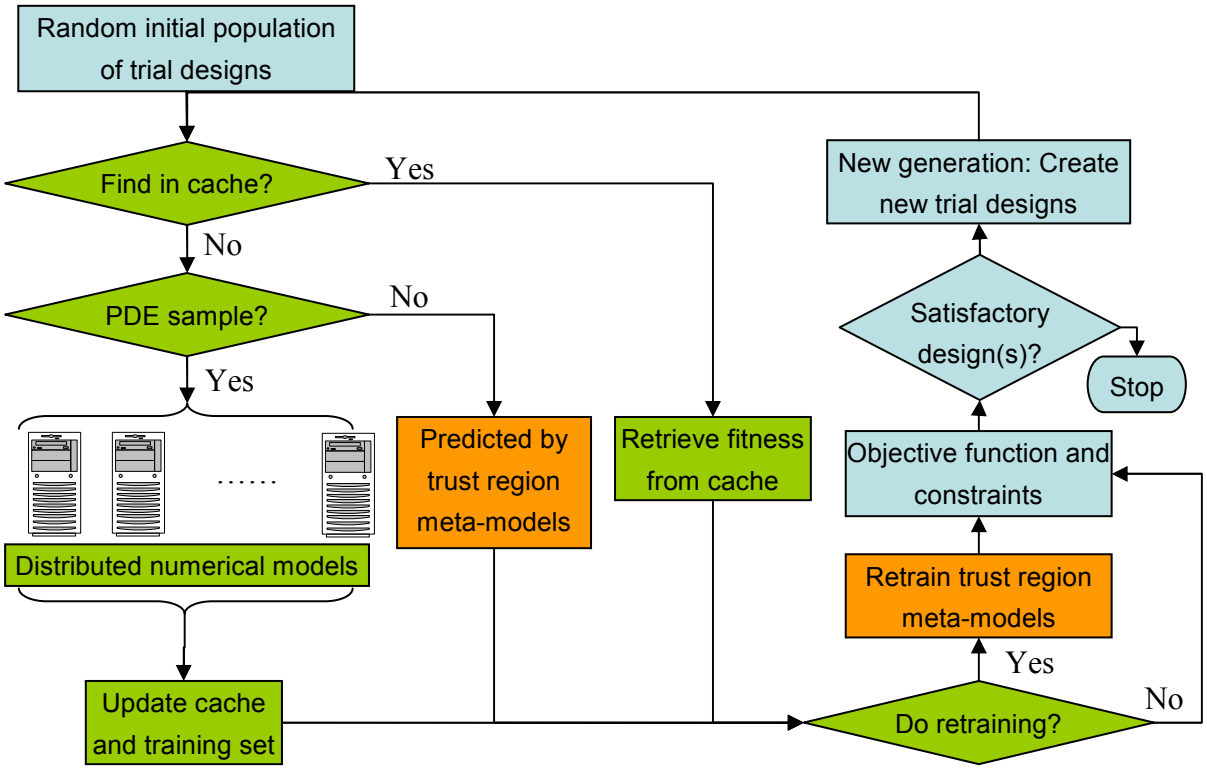

Figure 5.3. TRAMGA optimization framework.

The flowchart of TRAMGA is very similar to the AMGA flowchart (Chapter 4). The embedded prediction models, however, are replaced by trust region-based ANNs or trust region-based Support Vector Machines (SVMs). The training algorithm is also replaced by the trust region-based hierarchical training algorithm. A detailed description of the flowchart may be found in Chapter 4 (Figure 4.1).

5.2.4 Meta-models

Two machine learning models are used to construct hierarchical prediction models in this chapter. They are artificial neural networks and support vector machines. These machine learning models are chosen because they are capable of approximating complex non-linear functions, and they have been applied to a number of water resources

problems as shown in Chapter 2. The testing of SVMs in the dynamic GA framework also shows that the framework is a flexible design that different types of prediction models can be used.

Artificial Neural Network (ANN). ANNs have been introduced in Chapter 2. We used the two-layer ANNs in this chapter. The Levenberg-Marquardt backpropagation algorithm was used to train each individual ANNs in the hierarchical structure. Please also refer to Chapter 4 for detailed description of the training algorithm and ANN structure.

Support Vector Machine (SVM). SVMs have also been introduced in Chapter 2. In this study, we used LIBSVM, an SVM package obtained from researchers at the National Taiwan University (Chang and Lin, 2001). Initial runs showed that the radial basis function kernel within the package had a good prediction result, so this kernel function was used for this study.

5.3 Case Studies

Two water resources case studies were examined with the trust region-based hierarchical modeling approach. The first one is the Umatilla case study, described in Chapter 3. The second case study is a published nitrate concentration forecasting problem in the Upper Sangamon River, Illinois (Suen and Eheart, 2003). Using this previously published case study we can make a strict comparison among different meta-model prediction approaches. The following sub-sections give details of the TRAMGA setup for

the Umatilla case study (see Chapter 3 for other case study details) and discuss the Sangamon River case study in more detail.

5.3.1 Umatilla Case Study

The TRAMGA framework for the Umatilla case study periodically triggers the meta-model training processes at described in Chapter 4. In each training process, hierarchical regional models are constructed as described in Section 5.2. In previous work, it was found that there is only one small local region containing the global minimum, and the population of the GA gradually converges toward that region (Chapter 4, Yan and Minsker, 2006). Since the training set consists of trial designs sampled from the populations, it also converges toward the local region as the optimization progresses. In this study the cluster center was computed as the average of the most recently sampled solutions and a simple algorithm was used to construct the local training sets following two iterative steps: First the global model is constructed using the whole training set. Second at each lower level in the hierarchy (see Figure 5.1), the outside N points (the points further from the center) are discarded, and the reduced training set is used to construct a lower level local model. These two steps are repeated until the number of training points at the lower level is less than N. For this case study, N was set to a minimum of 400 to avoid small training sets and over-fitted meta-model models. Note that the above algorithm is specifically designed for uni-modal populations. For a multi-modal population, a clustering algorithm is required to identify local clusters, as described below for the Sangamon River case study.

The ANN and SVM models are trained to replace the flow and transport models. The internal ANNs in the trust region-based ANN models have the same structures as the described in Chapter 4 (Table 4.2). The SVM models are constructed similarly, which have the same inputs (pumping rates, pumping well locations) as the ANN models. Because each SVM can have only one output, five trust region-based SVM models are constructed to produce the five outputs (the maximum RDX and TNT concentrations in the fifth year, the GAC cost, the radios of the maximum RDX and TNT concentrations in the fourth year to the values in the fifth year). The SVM models use the radial basis kernel to transform the input data.

5.3.2 The Upper Sangamon River Case Study.

High nitrate concentration in water bodies is a potential risk factor for human health and U.S EPA has set 10mg/L as the maximum allowable nitrate concentration in drinking water. Predicting nitrate concentrations in rivers and streams is therefore important for water body management. The trust region-based modeling approach was also tested on a general nitrate prediction problem in the Upper Sangamon river basin. Suen and Eheart (2003) modeled nitrate concentrations in the Upper Sangamon river basin using a back-propagation neural network, a radius basis function neural network, a linear regression model and mechanistic water quality model, with a conclusion that back-propagation ANN outperformed the other three models. This case study was chosen for this research because the data set is easily accessible from the U.S Geological Survey, the size of the data set is medium so it can be tested with multiple models efficiently, and

the published results from Suen and Eheart can be used for strict comparisons.

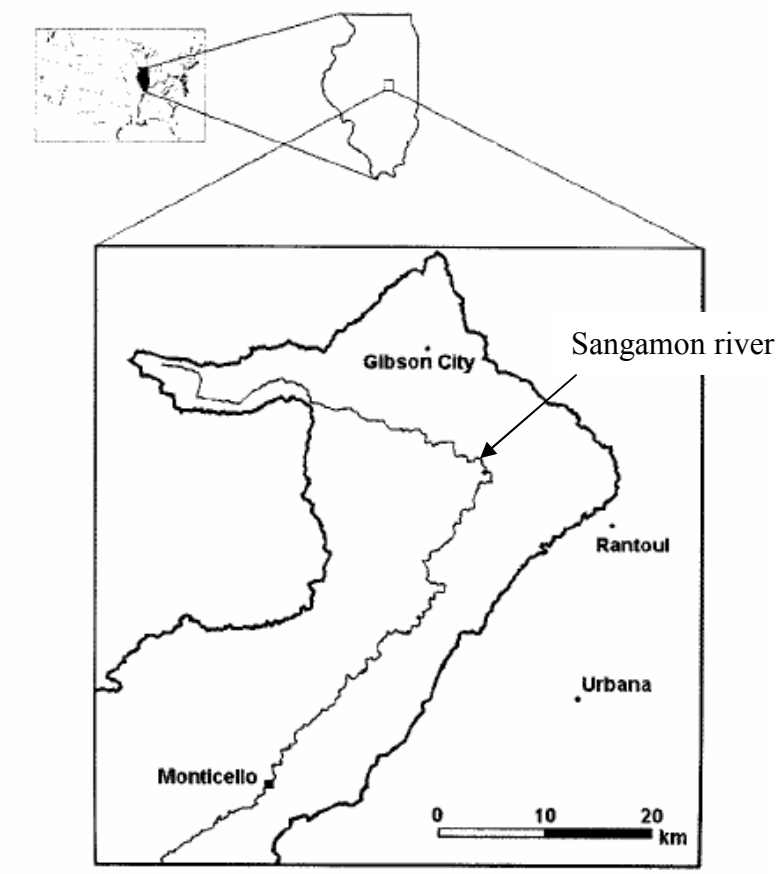

Figure 5.4 Planeview of Upper Sangamon River Basin, Illinois (Suen and Eheart, 2003)

Figure 5.4 shows the plan-view of the Upper Sangamon river basin. The basin is a typical mid-western river basin with a size of 1424 km^2. The land use of the basin is mostly agriculture, which also contributes to the nitrate concentration in the stream flow. High nitrate concentrations are usually observed between May and June when farmers apply fertilizers, and low concentrations are observed between August and October when the precipitation in the basin is high. A more detailed description is given by Suen and Eheart (2003).

Three weather stations and a gauge have been installed in the river basin. The three weather stations at Gibson City, Rantoul and Urbana collect daily rainfall and temperature data. The gauge at Monticello measures the river stream flow and nitrate concentration. The data collected from 1993 to 2000 at the weather stations and the gauge station are used as the data set for this work, as for Suen and Eheart (2003). The daily nitrate concentration at the Monticello gauge is the only model output. The model input parameters are the seven-day cumulative daily rainfall and daily highest temperature at the three weather stations, daily stream flow at the Monticello gauge, and the Julian date (which captures fertilizer application practices).

Suen and Eheart (2003) divided the available data set into a training set and a testing set. Two data separation approaches were tested. The first approach used the first half as the training set and the second half of the data as the testing set. The second approach used odd-year data as the training set and even-year data as the testing set. Their results showed that the odd-even separation approach produced better results. Therefore in this study, the odd-even separation approach was adopted.

The performance of the nitrate prediction models were evaluated on the test set based on several criteria. The Root Mean Square Error (*RMSE*) and the correlation coefficient (*CORR*) between the predicted nitrate concentrations and the measured concentrations were used to assess the models' prediction performance. The formula for computing *RMSE* and *CORR* are,

$$RMSE = \sqrt{\frac{1}{N}\sum_{1}^{N}(y_i - d_i)^2}$$

$$CORR = \frac{\sum_{1}^{N}(y_i - \bar{y})(d_i - \bar{d})}{\sqrt{\sum_{1}^{N}(y_i - \bar{y})^2 \sum_{1}^{N}(d_i - \bar{d})^2}} \tag{5.10}$$

where N represents the number of predictions, y is a predicted value, and d is an observed value. The performance was also assessed by the models' overall accuracy as defined by Suen and Eheart. The overall accuracy represents the fraction of data points that are correctly classified as above or below the 10 mg/l standard. For any incorrectly classified data points, the false positive frequency and false negative frequency were computed. Note that the false negative frequency has more negative consequence because it provides a false sense of public security, and may cause negative impact on public health or delay time-critical removal actions.

Suen and Eheart (2003) compared four different prediction models using the above performance criteria. These models included machine learning models (back-propagation neural networks and radial basis function neural networks), the soil and water assessment tool (SWAT, see Arnold et al., 1996) and multi-regression analysis (MRA). Among the prediction models, back-propagation ANN produced smaller RMSE and higher overall accuracy. In this study, four models were developed for comparison: a single ANN model having the same structure as the back-propagation neural network used by Suen and Eheart, a trust region-based ANN model, a single SVM model, and a trust region-based SVM model were tested. The SVMs here also used the radial basis

kernel. The predictions from these models were compared to the results published by Suen and Eheart.

The trust region-based model requires identifying local regions in the training data set so that local models can be established. Unlike the Umatilla case study, which has a single local optimal region of interest, the training set of the Sangamon case has multiple local regions. A K-means clustering algorithm was applied to the data set to identify local clusters, which were then used to construct local regions based on the identified clusters. One consideration in applying the clustering algorithm is to determine the clustering features. Clustering using only the input or output features can be found in past literature, for example Zhang and Govindaraju (2000). Clustering by the input features may divide the input space into subspaces, but this approach ignores the output function variation and may not produce the best smooth function in local regions. Clustering by the output features, on the other hand, may produce highly overlapping local regions, and subsequent predicting points are likely to be misclassified into incorrect regions (sub-models). In this study, both input and output features were used. The weighted input-output $(X, \boldsymbol{w}Y)$ space has a tuning parameter $\boldsymbol{w}$ to determine the proper combination.

5.4 Results

This section presents the results of applying the trust region-based models to the above test cases. First we present TRAMGA's performance when applied to the Umatilla

case study using both ANNs and SVMs as the internal prediction tools. Then we show the results of testing the trust region-based ANN and SVM models on the Upper Sangamon case study. Detailed results are presented in the following two sub-sections.

5.4.1 Umatilla Case Study Results.

The performance of TRAMGA was evaluated using both ANNs and SVMs as the internal prediction tools. In each case, TRAMGA was run with ten random initial populations (random seeds), and the results were compared to the corresponding AMGA (single meta-model modeling approach) runs that were also finished using the same ten initial populations. Note that for each initial population, TRAMGA and AMGA were identical in early generations until the training set in TRAMGA was large enough so that local models could be created. Note that in order to train a second level local model, TRAMGA has to wait until the training sets accumulated enough training points (*2*N*, See Section 5.3.1 for details). This identical generation interval was controlled by the minimum number of training points *N*. After this generation, the meta-models in TRAMGA are different from the meta-models in AMGA since they have local models established. Depending on the initial random population, TRAMGA required about 18~20 generations before launching into hierarchical model constructions.

The converged fitness values of both TRAMGA and AMGA using ANNs as prediction tools are shown in Figure 5.5. The horizontal axis shows the run number and the vertical axis shows the final converged fitness values. The figure indicates that most TRAMGA runs (except run number 6) converged to somewhat lower cost solutions than

those of AMGA runs. The exception in run 6 was likely caused by the GA's random effect.

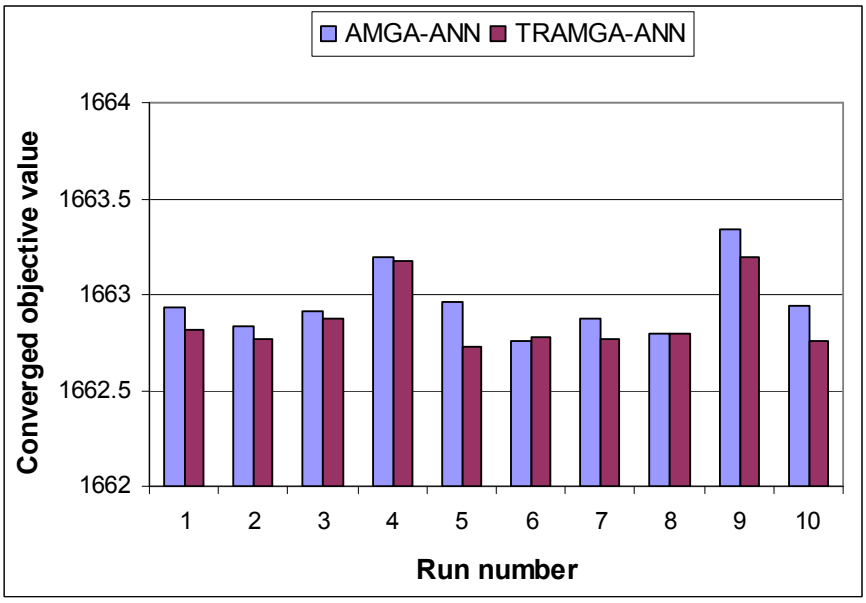

Figure 5.5. Converged fitness values of AMGA-ANN and TRAMGA-ANN for the Umatilla case study.

Figure 5.6 shows the required generations before the TRAMGA and AMGA runs converged. The figure indicates that the two algorithms had no significant difference in terms of required generations. This is because both AMGA and TRAMGA used a strict convergence criterion, which required that the standard deviation of the fitness values in a population be less than 0.0001. Although most TRAMGA runs found solutions with somewhat lower cost values, they still took roughly the same number of generations before the populations were completely dominated by the final solutions. In contrast, Figure 5.7 shows the generation number when the final solutions first appeared in the populations. After this generation the optimal solutions in the GAs were no longer improved, so this generation was used to quantify how fast the two algorithms identified

the optimal solutons. The figure shows that most TRAMGA runs required fewer generations to locate the final optimal solutions.

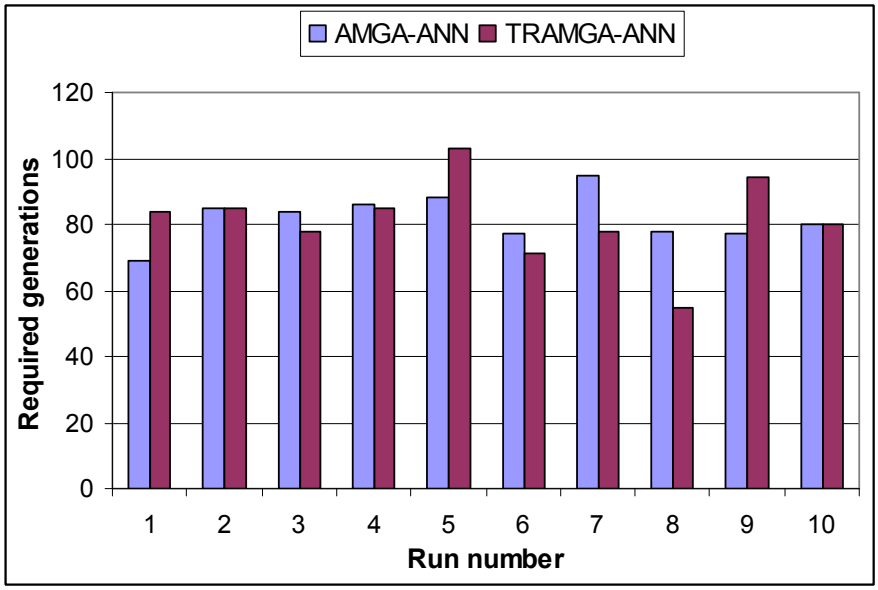

Figure 5.6. Converged generations of AMGA-ANN and TRAMGA-ANN for the Umatilla case study.

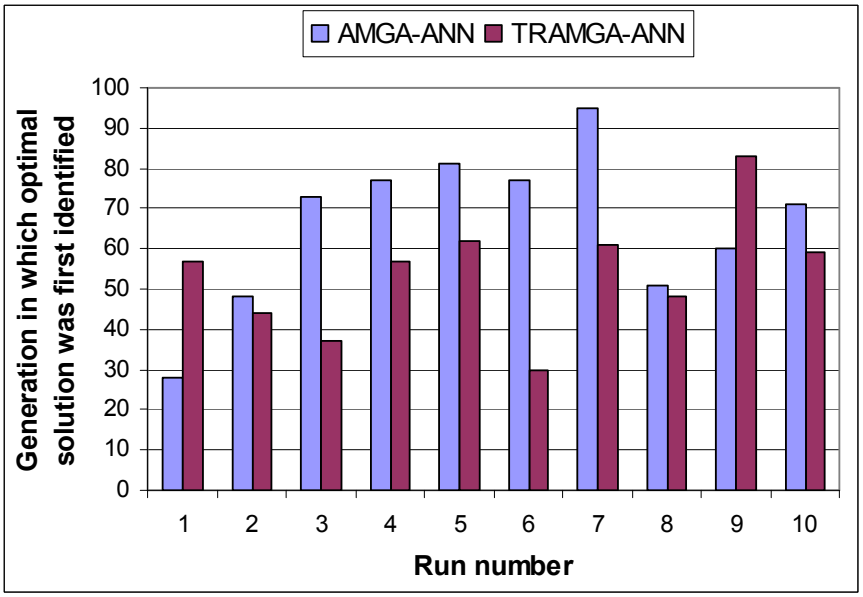

Figure 5.7. Minimum generation number when optimal solution was first identified for AMGA-ANN and TRAMGA-ANN.

The last comparison is of the error rate of the predicted fitness values. In each generation, after the sampled solutions were evaluated by the numerical models, the predicted mean square error of the fitness values was computed for these sampled solutions. Figure 5.8 shows the averaged fitness error of the sampled numerical-model-evaluated solutions in each generation. The sampling rate of AMGA adaptively decreases as the optimization progresses (see Chapter 4), therefore the sample size decreases from about 30 in early generations to as low as 1~2 in later generations. The error rate of TRAMGA (the dashed line) was lower than AMGA (the solid line) for most generations after hierarchical models were created. Note that in the adaptive GA framework, the meta-models tend to be over-optimistic for slightly infeasible solutions (Chapter 4, Yan and Minsker, 2006). In later generations, more solutions will be on constraint boundaries. These solutions were corrected by PDE sampling but they raised the overall error rate substantially due to high penalties associated with the constraints. This probably contributed to the higher error rate of TRAMGA in the last few generations, since it converged faster and the adaptive GA switched to the best sampling strategy, which is a more aggressive sampling method, to get rid of the slightly infeasible solutions and stabilize the population.

Similar results were observed when SVMs were used as the internal prediction tools. The converged fitness comparison is shown in Figure 5.9. Again, the TRAMGA-SVM runs converged to somewhat lower cost solutions than the AMGA-SVM runs when they started from the same initial populations. The convergence time for the two

algorithms again had no significant difference (Figure 5.10), and TRAMGA-SVM required fewer generations to identify the final solutions (Figure 5.11). The predicted fitness mean square errors of the meta-models are shown in Figure 5.12. Similar to the ANN runs, the TRAMGA-SVM runs had smaller prediction errors than the AMGA-SVM runs, albeit the improvement was smaller comparing to the ANN-based models.

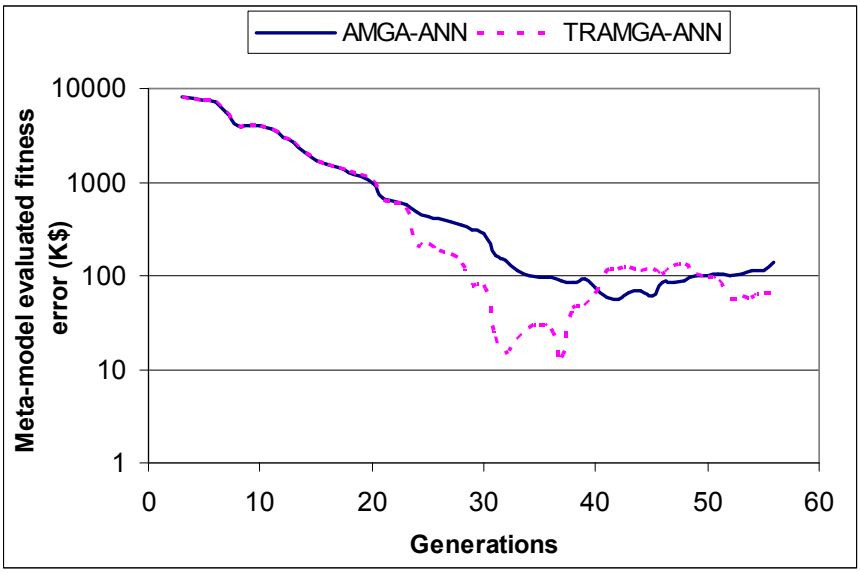

Figure 5.8. Fitness error comparison for the AMGA-ANN and TRAMGA-ANN runs.

Compared to the ANN model, the SVM model had less improvement in required generations for find the final solutions. Moreover, the improvement on fitness error when the trust region-based approach was used is smaller than the achieved improvement by the ANN models. This is probably because the response surfaces of SVMs are less flexible than the ANNs (Note that the SVMs use linear hyper-planes to approximate training data in the mapped high dimensional spaces rather than non-linear surfaces), so

the local models were not much different from the global model when SVMs were used in the hierarchical modeling structure (Haties et al, 2001).

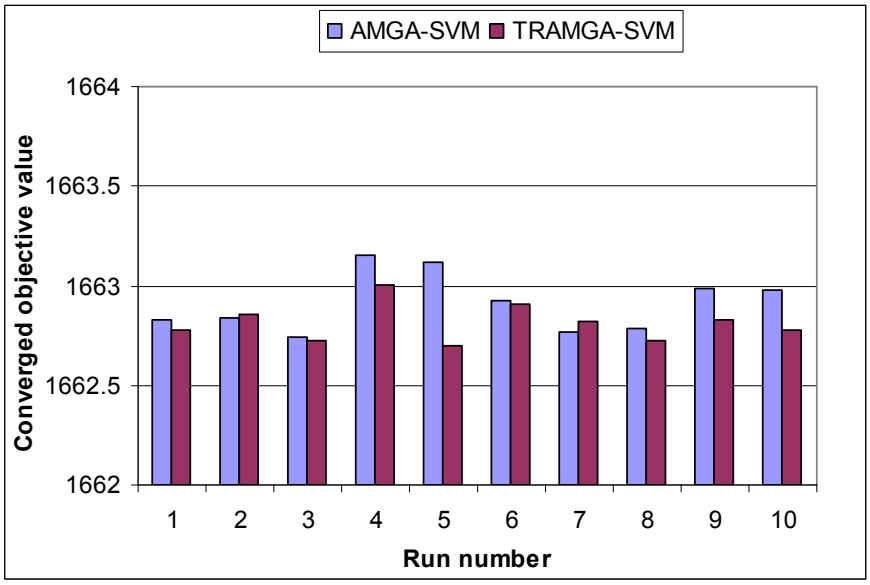

Figure 5.9. Converged fitness values of AMGA-SVM and TRAMGA-SVM runs for the Umatilla case study.

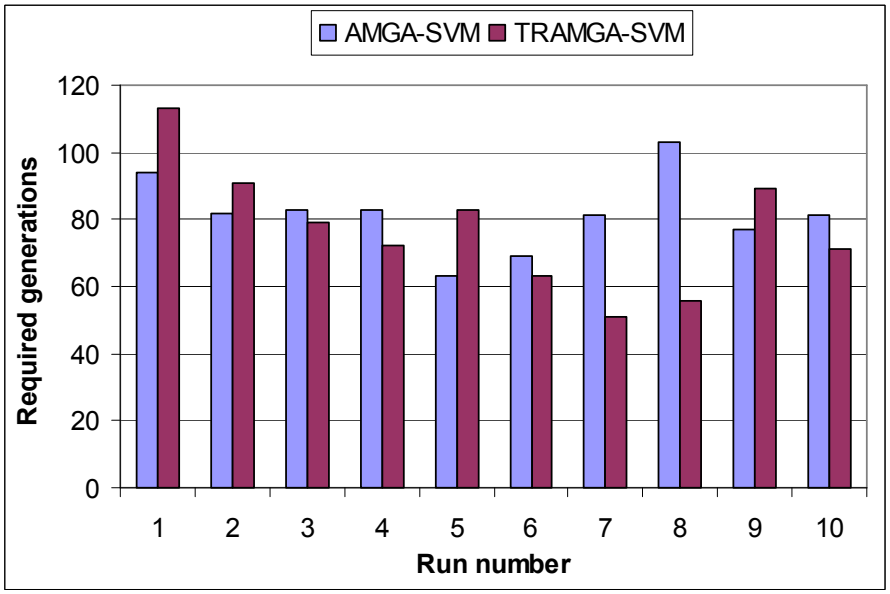

Figure 5.10. Converged generations of AMGA-SVM and TRAMGA-SVM for the Umatilla case study.

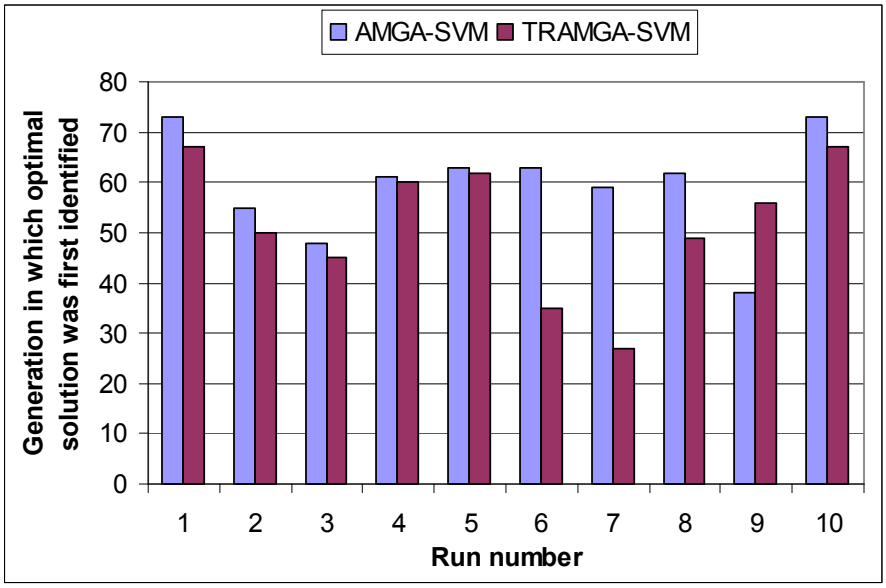

Figure 5.11. Minimum generations of AMGA-SVM and TRAMGA-SVM for identifying the optimal solutions for the Umatilla case study.

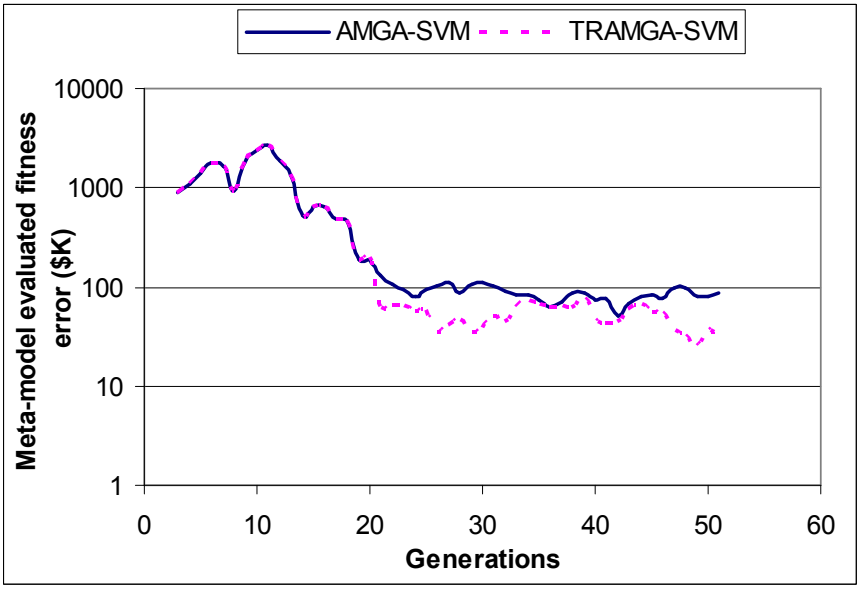

Figure 5.12. Fitness error comparison for AMGA-SVM and TRAMGA-SVM runs for the Umatilla case study.

5.4.2 Sangamon Case Study Results.

The Sangamon case study was also tested with both ANNs and SVMs. Unlike the Umatilla case, the Sangamon case data set had multiple local clusters, hence the K-means clustering algorithm was used to identify the local clusters.

Figure 5.13 shows the clustering performance in terms of the silhouette value as the number of clusters increases from two to five. The silhouette value is formulated as,

$$S_i = \frac{\min(b_i, 2) - a_i}{\max\left[a_i, \min(b_i, 2)\right]} \tag{5.11}$$

where S_i is the silhouette value for point i, a_i is the average distance from point i to other points in the same cluster, b_i is the average distance to the points in the closest cluster. The silhouette value measures how similar a point is to points in its own cluster relative to points in other clusters, ranging from -1 to +1 (Kaufman and Rousseeuw, 1990). High silhouette values indicate that points are likely classified into correct clusters. Otherwise, low or negative silhouette values indicate that points are probably misclassified. The silhouette values in figure 5.13 show that when the cluster number goes beyond two, a small, low-quality cluster with a long negative tail appears. This indicates that many points are misclassified due to the negative silhouette values. Therefore a cluster number of two is a proper choice for splitting the training set into local regions.

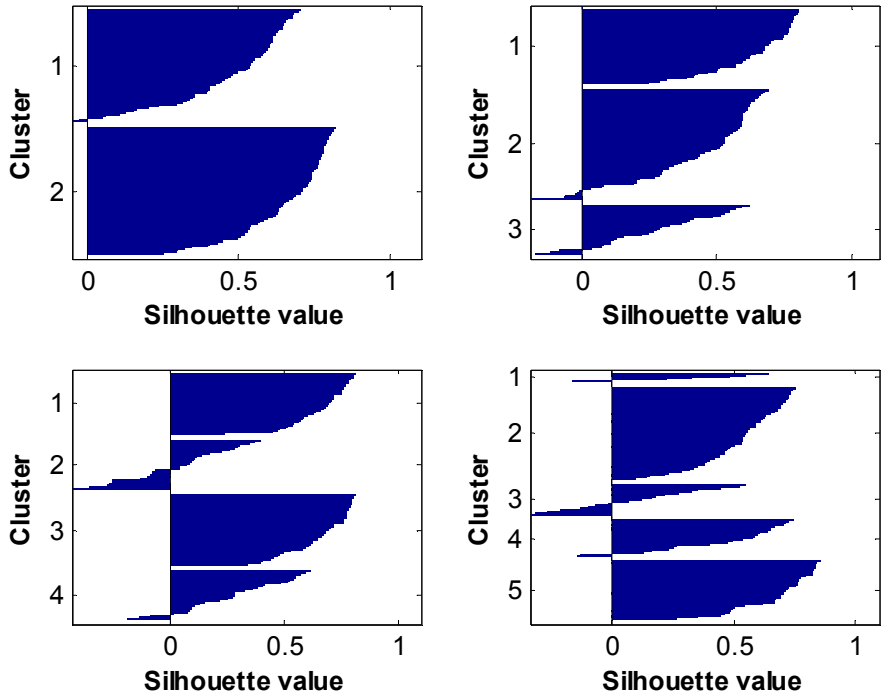

Figure 5.13. Cluster silhouette values when cluster number is increased from 2 to 4 for the Sangamon data set.

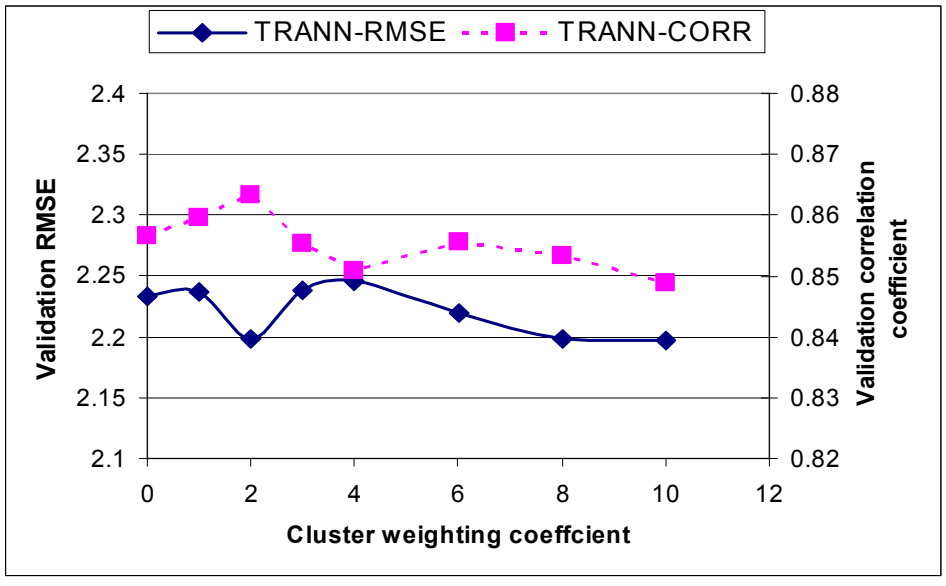

Figure 5.14. TRANN performance as cluster weighting coefficient increases for the Sangamon case.

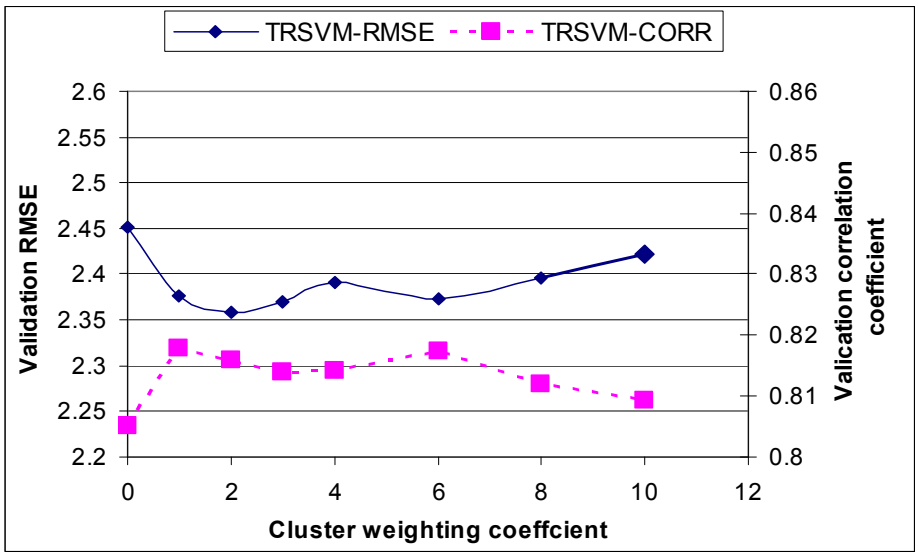

Figure 5.15. TRSVM performance as cluster weighting coefficient increases for the Sangamon case.

After the cluster number was chosen, the input-output weighting coefficient was tested to determine the best value. Figure 5.14 shows the prediction performance of the trust region-based ANN model as the weighting coefficient **w** was increased from 0 to 10. Note that when **w** is 0, the clustering algorithm clusters only in the input feature space. On the other hand, when w is as large as 10, the output feature overwhelms the input features and the clustering algorithm effectively clusters only in the output feature space. Figure 5.14 indicates that the *RMSE* (the solid line) initially decreases as **w** increases from 0. Conversely, the correlation coefficient (the dash line) initially increases as **w** increases from 0. Note that a low *RMSE* indicates a low prediction error and a high *CORR* indicates that the predictions and the observations are highly correlated. When **w** is set to close to 2, the best performance is obtained. At this point the *RMSE* drops to a local minimum and the correlation coefficient reaches its maximum value. Although the *RMSE* decreases again after **w** is larger than 5, the correlation coefficient is lower for

larger weighting coefficients. A similar performance variation was also observed with the trust region-based SVM model, as shown in Figure 5.15. Again, the lowest *RMSE* and the highest correlation coefficient are obtained when **w** is set to 2. Therefore in the following test runs, **w** is set to the optimal value of 2.

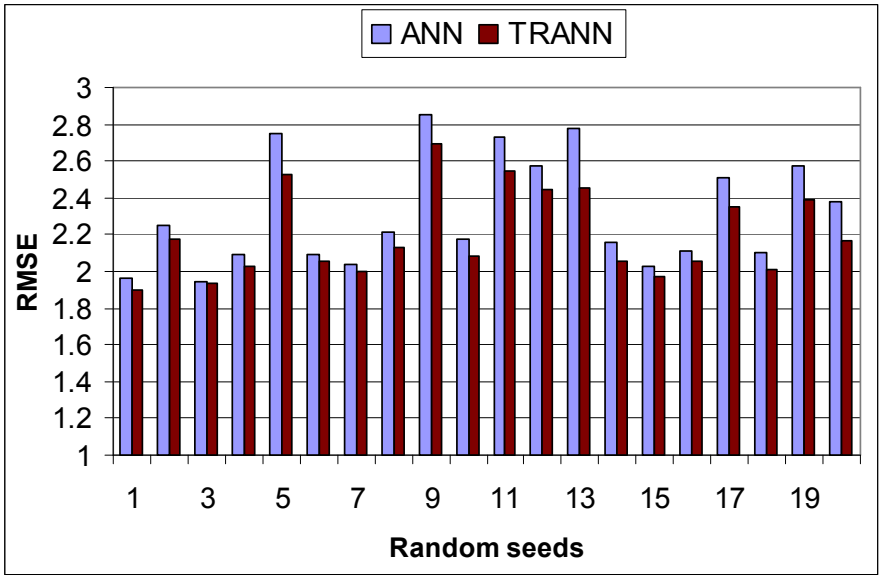

Figure 5.16. TRANN prediction errors for the Sangamon case when tested with multiple random seeds.

The Levenberg-Marquardt training algorithm may converge to different local minima, depending on the random seed selected to initialize the ANN weight matrix. To ensure that the comparison is not affected by this phenomenon, both the single and trust region-based ANNs were trained with 20 random seeds. The prediction *RMSEs* and the overall accuracies (see Section 5.3.2 for their definitions) are shown in Figures 5.16 and 5.17, respectively. In the 20 test runs, almost all of the trust region-based ANN models had smaller prediction errors than the corresponding single ANN models. In term of the overall accuracy, the trust region-based ANN model had higher overall accuracy than the

single ANN model for most random seeds. Therefore, the trust region-based ANNs can achieve better prediction accuracy than the single ANNs.

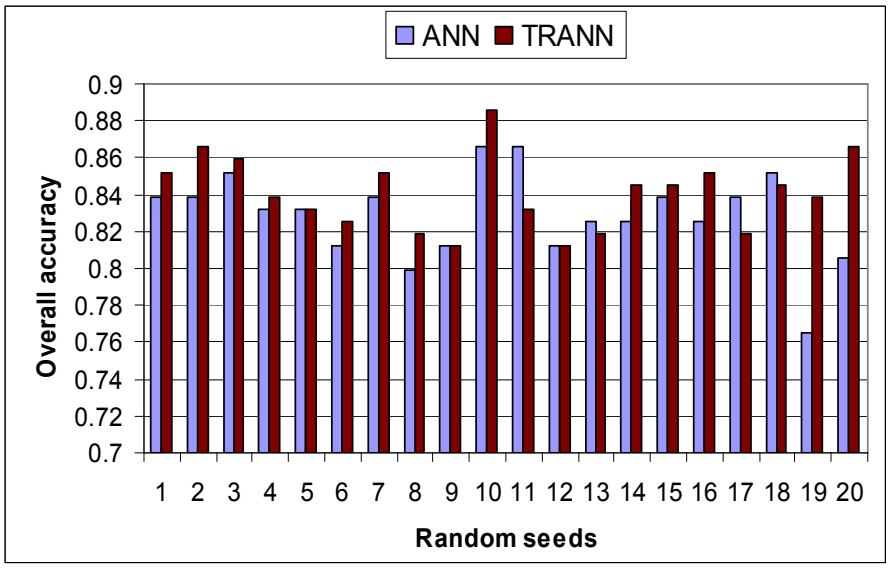

Figure 5.17. TRANN prediction overall accuracies for the Sangamon case when tested with multiple random seeds.

A similar comparison was performed using the single SVM model and the trust region-based SVM model, as shown in Figure 5.18 and Table 5.1. Since the SVM training algorithm minimizes a convex function, a unique solution is always obtained regardless of the random seeds. Compared to the single SVM model, the trust region-based SVM model had smaller *RMSE*, higher correlation coefficient, higher overall accuracy, higher false-positive frequency (high false-positive is preferred, Section 5.3.2). Hence the trust region-based SVM model improved the performance on all the performance criteria.

Figure 5.18 and Table 3 show the best results from all models, including the ANN, SWAT and MRA models published by Suen and Eheart (2003). The ANN model is the

best model reported by Suen and Eheart. Note that the radius basis function ANN (Suen and Eheart, 2003) is not as good as the back-propagation ANN, hence we only compare our models to their back-propagation ANN. Comparing the six different prediction models, the SWAT model had the highest RMSE and lowest overall accuracy, which was probably due to the inaccurately evaluated parameters that are typical when estimated from empirical data (Suen and Eheart). The MRA model had better prediction accuracy than the SWAT models, but the performance was still poor in general. The SVM model had good performance in terms of overall accuracy, but the false-negative rate was higher than other methods. While the trust region-based SVM model had improved performance over the single SVM model on all of the assessment criteria, the false-negative rate was still high. The ANN model attained much lower prediction error and much higher correlation coefficient than the above models. When the trust region-based ANN model was used, further improvement was achieved. The trust region-based ANN improved the overall accuracy by 1.6% and RMSE by 3.2% over the second best model (single ANN model) but with slightly degrading on false-positive frequency. Compared to the best model reported by Suen, the trust region-based model improved the overall accuracy and RMSE by 2.4% and 7.4%, respectively. In general, the best trust region-based ANN model attained the lowest RMSE, the highest correlation coefficient, high overall accuracy (85%) and low false negative rate (22%).

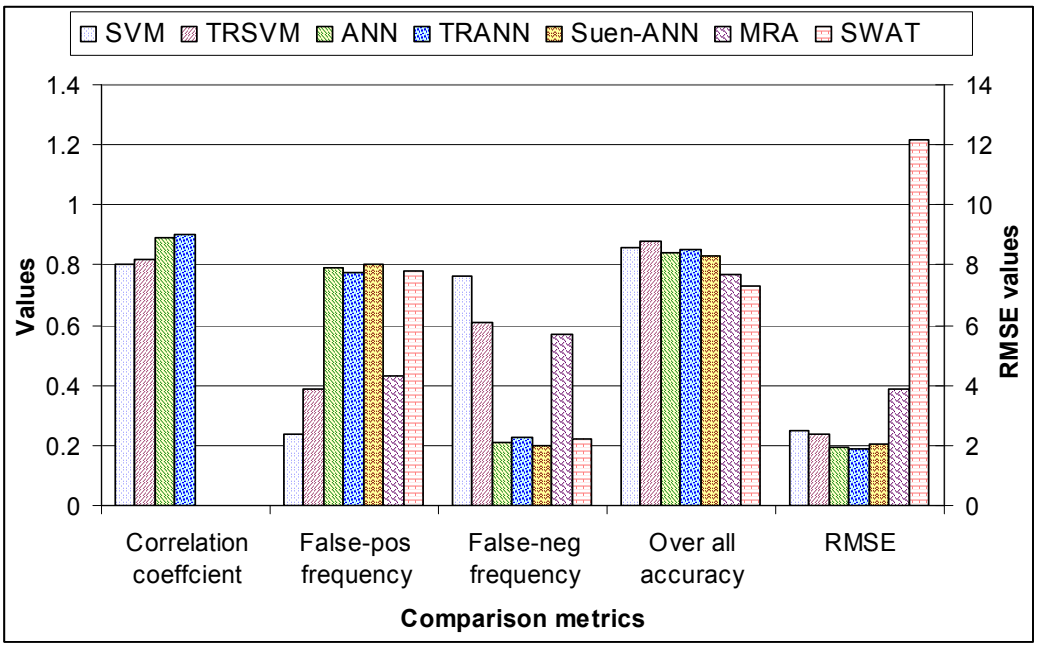

Figure 5.18. Performance comparison of ANN, SVM, TRANN, TRSVM, MRA and SWAT models for the Sangamon case.
(MRA and SWAT model results are from Suen and Eheart, 2003).

Table 5.1. Comparison of ANN, SVM, TRANN, TRSVM, MRA and SWAT performance

Model	Correlation coefficient	False-positive frequency	False-negative frequency	Overall accuracy	RMSE
SVM	0.801	0.238	0.762	0.859	2.472
TRSVM	0.819	0.389	0.611	0.879	2.357
ANN	0.892	0.792	0.208	0.839	1.959
TRANN	0.900	0.773	0.227	0.852	1.897
Suen-ANN		0.800	0.200	0.832	2.05
MRA		0.429	0.571	0.771	3.901
SWAT		0.778	0.222	0.728	12.151

(MRA and SWAT model results are from Suen and Eheart, 2003).

5.5 Conclusions

Data-driven meta-model models are valuable to achieving realistic solutions to complex water resources problems when conceptual models are incapable of fully

capturing the characteristics of underlying physical processes or the conceptual models are too expensive to be finished within required time limit. In this chapter, a divide-and-conquer approach that hierarchically combines a global model with multiple detailed local models in a trust-region framework was proposed. The global model is a general prediction model and the local models are independent sub-models established on subspaces. These models are selected by a trust region test for predicting given input features, and the models outputs are mixed by a gating network and a soft mixing function to produce the final prediction. This modeling structure can adopt different types of prediction tools, including artificial neural networks and support vector machines in this study.

The trust region-based modeling structure was tested on two case studies. The first case study used the approach within the TRAMGA framework for solving the Umatilla case study. For both ANN and SVM prediction tools, the results show that TRAMGA gave more accurate solutions with somewhat lower cost values than the original adaptive framework. Moreover, the trust region based modeling approach identified the optimal solutions more rapidly, and attained lower prediction errors for most generations. Compared to the SVM models, TRAMGA had more improvement over AMGA when ANNs were used. The successful optimization results using ANNs, SVMs, trust region-based ANNs and trust region-based SVMs also indicate that the adaptive GA framework is a general design that can incorporate different types of meta-models.

The trust region-based modeling approach was then tested on a published nitrate prediction case study in the Upper Sangamon river basin again using both ANNs and SVMs. The trust region-based ANN model was tested on multiple random seeds and a clear improvement in both prediction error and overall accuracy was observed, although the false-positive and false negative frequency degraded slightly. The SVM models converged to an unique predictor and the trust region-based SVM model achieved better performance on all assessment criteria than the single SVM model. All of these models produced better results than the MRA model and SWAT model reported by Suen and Eheart(2003), and the trust region-based ANN model improved the overall accuracy by 2.4% and RMSE by 7.4% over the best model (ANN) reported by Suen and Eheart. Overall, the trust region-based ANN model was probably the best choice among all approximation models since it had the lowest prediction error, the highest correlation coefficient, 85% overall accuracy and only 22% false negative rate.

The results across the two case studies also indicate that although the trust region-based method can improve the prediction accuracy, model selection is still the most important factor in determining the overall prediction performance. Different models have different abilities in learning complex functions from different training data sets. For a specific data set, the trial and error test is generally required to determine the most appropriate prediction model. In the future, a mixture of different prediction models can be explored. This approach enables adopting the best local models in different local

regions within a hierarchical structure, and therefore has the potential of further improving the prediction performance.

CHAPTER 6

OPTIMIZING GROUNDWATER REMEDIATION DESIGNS UNDER UNCERTAINTY

6.1 Introduction

The optimization methods proposed in Chapter 4 and 5, although computationally competent, are deterministic approaches that assume fixed parameter configurations during the optimization processes. This assumption can be unrealistic in real world groundwater remediation problems because our knowledge of the involved parameters and the initial conditions is often incomplete and inaccurate. The existence of natural randomness in remediation problems further shakes our confidence in the solutions found by deterministic optimization methods. As a result, uncertainty analysis that can characterize parameter uncertainty is desired in optimization processes to identify robust solutions with a preferred confidence level. In groundwater remediation problems, hydraulic parameter uncertainty is a typical uncertainty source. The parameter uncertainty is translated into simulation model prediction uncertainty through the involved flow and transport models. Therefore, characterizing the prior parameter uncertainty results into the posterior prediction distributions on which constraints can be enforced.

In the past literature, two types of statistical optimization approaches have been applied in most water resources management problems with uncertainty. The first approach is the chance constrained optimization approach. Examples of applying the

method include Wagner and Gorelick (1987), Tiedeman and Gorelick (1993), and Chan (1994). This method uses the prior parameter distributions to formulate the posterior joint distributions of constraints. Then deterministic equivalents of the constraints can be obtained by enforcing constraints on the posterior joint distributions to a desired success probability. One disadvantage of the chance constrained optimization is that it requires computing the posterior joint distributions, which can be difficult to obtain when complex simulation models are involved in evaluating constraints, such as for groundwater remediation. As an alternative, scenario-based optimization approach can be used, which involves generating a significant number of equally likely parameter realizations from the prior distributions. Each realization is a configuration of the uncertain parameter values. During the optimization process, Monte Carlo-type sampling technique can be used to sample realizations so that the design solutions can be evaluated over multiple parameter realizations. Examples of the methods include Wagner and Gorelick (1989), Chan (1993), and Morgan et al (1993). A typical scenario based stochastic optimization approach, which is adopted in this research, is the Noisy Genetic Algorithm (Noisy-GA).

Noisy-GAs operate like traditional GAs except that the algorithms use sampling techniques to estimate the objective function expectation by sampling uncertain parameters. The sampling technique makes the GA less likely to converge to over-optimistic solutions that can easily fail the constraints by small perturbations of the parameters, but can suffer from severe computational overhead when used with

computationally intensive simulation models because the underlying Monte Carlo sampling evaluates each trial design over multiple parameter realizations.

In this chapter, an approach for addressing this issue is proposed, which combines neural network meta-models with a dynamic noisy GA. This framework is an extension to the Adaptive Meta-model GA (AMGA), but significant modifications have been made so that the framework can be applied in an uncertain environment where the learning data have sampling noise. The detailed methodology is described in the next section (6.2). Following the methodology two remediation case studies are discussed in Section 6.3. Then the results of applying Noisy-AMGA to the two case studies are presented in Section 6.4. Finally the chapter is concluded in Section 6.5.

6.2 Methodology

Since the Noisy-AMGA method is an extension of AMGA, the method shares similar characteristics in updating the embedded meta-model surrogates and distributing the simulation model evaluations. Noisy-AMGA, however, has several new features in order to handle uncertainty in the framework. In section 6.2.1 below, an overview of Noisy-AMGA is given. Subsequent subsections then introduce the technical details, which include a summary of the standard Noisy-GA approach (6.2.2), a new incremental Latin Hypercube sampling approach (6.2.3), meta-model training based on statistical decision theory (6.2.4), the extended archiving cache (6.2.5), and a probabilistic selection strategy (6.2.6).

6.2.1 Noisy-AMGA Framework

Figure 6.1 shows the flowchart for the noisy adaptive meta-model genetic algorithm (Noisy-AMGA), which resembles the AMGA framework proposed in chapter 4 but with modifications to incorporate parameter uncertainty. Here again we used ANNs as the internal meta-models because ANN-based AMGA has been extensively tested on the two case studies in previous chapters, and they are more flexible in incorporating different model structures than SVMs. As with AMGA, a majority of the trial designs are evaluated by adaptively updated meta-models. Only a few trial designs are selected for numerical simulations using similar sampling selection strategies as presented in Chapter 4 and Yan and Minsker (2006). Each of the selected trial designs is then evaluated over multiple hydraulic conductivity realizations sampled by a new incremental Latin Hypercube algorithm. These evaluations are executed in parallel in an underlying distributed computing infrastructure. The simulation results are then used to update the cache and retrain the meta-models. The optimization problem may be stated quantitatively as,

$$\begin{aligned} &\textit{Min } E[COST] \\ &\textit{S.T.} \\ &\quad \textit{Hydraulic and cleanup constraints} \\ &\quad \textit{Reliability} = P(\textit{hydraulic and clean constraints are met}) \geq \alpha \end{aligned} \tag{6.1}$$

The difference between AMGA and Noisy-AMGA is that the latter minimizes the fitness expectation across sampled realization scenarios rather than fitness based on a single fixed realization. Besides the ordinary hydraulic and cleanup constraints, a reliability constraint is also included in Noisy-AMGA so that the algorithm can converge

to solutions with a desired level of reliability α. Here the reliability denotes the percentage of realizations that satisfy all constraints. Furthermore, a cache is used in Noisy-AMGA as an archive in which evaluated chromosomes and their associated realization characteristics are stored. When a trial design is selected for numerical model evaluations multiple times, the expected fitness is improved by combining current sampled realizations with any previously sampled realizations in the cache. Further details on Noisy-AMGA features are given below.

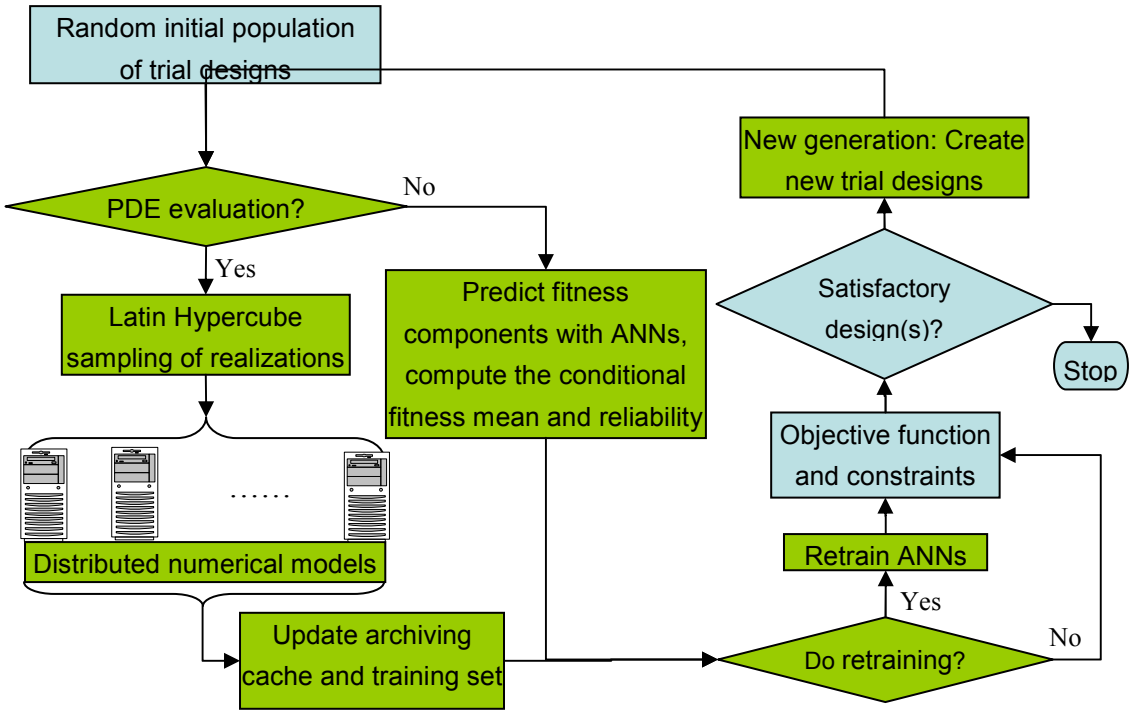

Figure 6.1. Noisy AMGA optimization framework

6.2.2 Noisy Genetic Algorithm

Noisy Genetic Algorithm (Noisy-GA) is a stochastic optimization algorithm introduced by Fitzpatrick and Grefenstette (1988). Noisy-GAs use ordinary GA operators but they can optimize in a noisy or uncertain environment in which the fitness evaluation

of trial designs is subject to perturbations from the use of noisy data (hydraulic conductivity in this research). In Noisy-GAs, trial designs are evaluated by a sampling function to compute the average fitness. This averaging feature absorbs parameter noise by averaging fitness samples drawn from multiple plausible realizations. The averaged fitness values converge to the actual expectation through the central limit theorem as the number of samples increases. Therefore, Noisy-GAs actually minimize the conditional fitness expectation given the trial designs and the uncertain parameters.

In Noisy-GAs, multiple parameter realizations are sampled to compute the fitness expectations. The number of realization samples is critical for accurately estimating the conditional fitness expectations. Although Monte-Carlo-type sampling techniques can be used to obtain accurate estimation results from a probability distribution, extensive sampling is not required in Noisy-GAs (Miller and Goldberg, 1996). In 2003, Gopalakrishnan et al found that a low sampling number (as low as 5 samples per solution) was able to identify highly reliable solutions using the hypothetical remediation case study described in Chapter 3. This fixed low sampling number, however, is not necessarily appropriate in a more realistic case where the uncertain parameters are likely distributed in a much wider range. In the AMGA framework, the error of the neural-network meta-models can also increase the noise and could lead the GA to incorrect solutions. Hence the design and implementation of effective sampling in Noisy-AMGA is even more important.

6.2.3 Incremental Latin Hypercube Sampling

When parameter realizations are sampled for a trial design, numerical models must be evaluated to compute the fitness value for each of the sampled parameter realization. Since the numerical model evaluations dominate the computational time, the computational complexity of Noisy-GAs is proportional to the sampling number N. In reality, a low sampling number is always preferred to speed up the computation. The low sampling number, unfortunately, may produce highly biased estimates of the unknown fitness expectations when simple sampling techniques (such as pure random sampling) are used. In order to produce less biased estimations, a more efficient sampling approach, an incremental Latin Hypercube Sampling (LHS), was utilized in this research.

LHS is a stratified sampling technique which divides a random distribution into non-overlapping equal intervals of the cumulative probability (Mckay et al., 1979). LHS is able to produce more accurate fitness expectation estimates than the simple random sampling because it guarantees that the cumulative probability range is equally covered. The original LHS algorithm is a memoryless sampling algorithm that does not account for previously sampled realizations if the same trial design is chosen for LHS sampling multiple times. Suppose a trial design undergoes LHS twice, and each time N realizations are sampled. Although each set of N realizations is equally divided into N intervals, the overall $2N$ realizations are not necessarily equally divided into $2N$ non-overlapping intervals because the second set of N samples may be close to or overlap with the first set of N samples. Because the Noisy-AMGA implementation uses an archiving cache to

store previously sampled realizations, it is desirable to utilize the available history information to improve the LHS sampling efficiency.

The improved LHS algorithm, called incremental LHS, works as follows: Suppose a trial design was already sampled with M parameter realizations. If the trial design is to be sampled again for N additional realizations, the M realization information is first delivered to the incremental LHS as an array. The incremental LHS then divides the sampling region into (M+N) equal intervals. Obviously, there are at least N empty intervals that contain no sampled realizations. The incremental LHS randomly selects N intervals (if there are more than N empty intervals) and samples a realization from each of the N intervals. Note that the algorithm is not an exact incremental LHS, which usually requires more memory or more computation, e.g. Pleming and Manteufel (2005). It is possible that some intervals may not be sampled at all when there are more than N empty intervals. Nevertheless, the algorithm is a simple and efficient approximation of the exact incremental LHS.

6.2.4 Extended Archiving Cache

In the original AMGA design, a cache is used to save evaluated chromosomes and the associated fitness values. The cache bypasses simulation evaluations if the saved chromosomes appear again in later generations. This technique works poorly in Noisy-AMGA because each chromosome is sampled with a small number of parameter realizations. Even if the same chromosome appears in later generations, it is unlikely to be sampled with the same realizations. The true function of the cache in Noisy-AMGA is

to store previously sampled realization information to improve the expectation estimation of the incremental LHS algorithm.

The cache in Noisy-AMGA is an extended archiving cache that saves the evaluated chromosomes, their associated realization information and their fitness values. If a saved chromosome is selected for numerical model evaluation again, the cache provides the *M* archived realization information so that the incremental LHS can sample additional *N* evenly covered realizations. After the fitness values associated with the new realizations are evaluated, the combined *(M+N)* samples are used to estimate the fitness expectation. The archiving technique has the advantage that the estimation of fitness expectation is progressively improved as the GA moves into the local region of the optimal solution, since the near-optimal solutions are likely sampled multiple times and reliable estimations can be obtained from the large number of samples. A similar technique was also used by Singh and Minkser (2003) and Singh (2004), but in their study only the parent and children samples were archived instead of archiving all historical samples.

A similar AVL tree structure described in Chapter 3 is used as the archiving cache, but the AVL tree is modified so that each node can store the sampled realization information. Details of the AVL tree structure was described in Chapter 3.

6.2.5 Meta-model Training Based on Statistical Decision Theory

In a noisy data set, a given input decision vector produces multiple outputs due to the parameter sampling technique. These sampled outputs form a posterior distribution,

and the training set describes the posterior distribution at different input locations. If a meta-model is trained on such a noisy data set, the meta-model would have a set of outputs to learn for each input vector. Statistical decision theory can be used to explain why a meta-model can effectively learn a population quantity from the noisy data set, as described below.

Supervised learning algorithms are used to train machine learning models when both inputs and observations are available. The training algorithms try to find appropriate response functions mapping inputs to observed outputs. The objective of a supervised training algorithm is to minimize a loss function that describes the discrepancy between the observed and the predicted outputs. The most popular loss function used in practice is the squared error loss function,

$$L(y, f) = \int \left(y - f(\mathbf{x})\right)^2 p(y \mid \mathbf{x}) dy = E\left[\left(y - f(\mathbf{x})\right)^2 \mid \mathbf{x}\right] \tag{6.2}$$

where $\boldsymbol{x}$ is an input vector, $f(\boldsymbol{x})$ is the trained predictive function, and y is an observed output. The squared loss function is typically minimized by a training algorithm, which leads to an optimal regression function approximating the conditional expectation of the prediction values (Hastie et al. 2001). Suppose the optimal solution is denoted as

$$f^*(\mathbf{x}) = \underset{for\ all\ f(x)}{\arg\min}\, E\left[\left(y - f(\mathbf{x})\right)^2 \mid \mathbf{x}\right] \tag{6.3}$$

This optimal function $f^*(\mathbf{x})$ can be derived by taking the derivative of equation 6.3, which has a solution of,

$$f^*(\mathbf{x}) = E\left[y \mid \mathbf{x}\right] \tag{6.4}$$

Equation 6.4 implies that the optimal response function (target function) is actually the conditional expectation given the input vector **x** and the random sampling noise. This equation indicates that the training algorithm of any supervised learning model selects the best approximation to this ideal expectation within the restricted class of functions defined by the meta-model structure and the finite training data. Since in Noisy-AMGAs the conditional expectation is exactly the fitness function being minimized, the decision theory indicates that the trained meta-models can be used as fitness meta-models in Noisy-AMGAs.

In practice, meta-models seldom achieve the ideal target functions. This is because a learning function is restricted by the meta-model's internal structure, and the learning function can only learn from a finite sampling of the unknown population distribution. A meta-model's prediction error is always higher then the Bayes error (the lowest error that can be achieved by the ideal response function). It is usually difficult to quantify a meta-model's error bounds without introducing substantial computational overhead (eg. using bootstrapping or Markov Chain Monte Carlo simulation [Hastie et al. 2001 and Neal, 1996]). If the meta-model's structures are carefully designed and there are significant amounts of training data, however, the meta-model's error can be neglected. In this research, we assume that the meta-models approximate the target functions very well. This is a legitimate assumption because GAs are statistical algorithms that can tolerate small errors in the fitness functions. The assumption also

simplifies Noisy-AMGA's selection operator, which will be shown in the next sub-section.

The fact that supervised learning models learn population quantities in noisy environments implies that global models are more resistant to noise than local models. Local models are more capable of learning local details, but may learn too much noise when local training data sets have uncertainty. Especially in local regions, the trained local models are likely to have high prediction variance that may cause degraded performance. In this study, only the global meta-model was used to avoid these issues.

6.2.6 Probabilistic Selection in Noisy-AMGA

Selection is a key operator in Noisy-AMGA for handling noisy data. The popular selection operator, tournament selection, requires comparing different trial designs so that the best one can be selected for mating. A simple approach used in Noisy-GAs is to directly compare the mean fitness values after the trial designs are sampled with multiple parameter realizations. This approach does not account for the standard deviations of the sampled fitness values and may produce erroneous results when the mean fitness values are close. A more realistic approach that compares the trial designs in a probabilistic model was proposed by Huges (2001), who assumed that the averaged fitness values follow normal distributions. The fitness difference of two trial designs is then the difference of two normal distributions, which is again a normal distribution,

$$N = \left(\frac{(\overline{X}_A - \overline{X}_B) - (\mu_A - \mu_B)}{\sqrt{\sigma_A^2 + \sigma_B^2}} \right) \tag{6.5}$$

where $\overline{X}_A$ and $\overline{X}_B$ are the sampled means of fitness values of trial design A and B, μ_A and μ_B are the true means of the trial designs, and σ_A^2 and σ_B^2 are the variance of the trial designs.

Equation 6.5 can be used to construct a hypothetical test to determine the probability of trial design A's fitness being less than trial design B's fitness. The equation assumes that the true standard deviations of the fitness values are known or can be evaluated accurately. This assumption is usually difficult to satisfy in practice, where the standard deviations are estimated from limited samples. Singh (2003) proposed a more general probability model that does not require a known standard deviation assumption. The statistical approach uses a two sample student test to compare the fitness of two solutions, each having different numbers of realization samples. For two given fitness sample sets of size *m* and *n*, the following *T* distribution can be constructed,

$$T_{n+m-2} = \left(\frac{(\overline{X}_A - \overline{X}_B) - (\mu_A - \mu_B)}{\sqrt{\frac{(n-1)s_A^2 + (m-1)s_B^2}{n+m-2}\left(\frac{1}{n} + \frac{1}{m}\right)}} \right) \tag{6.6}$$

where *T* is a student distribution with *n+m-2* degrees of freedom, and *n* and *m* are the sample numbers that are used to compute the sampled means ($\overline{X}$) and standard deviations (*s*).

Singh (2003) used the above test to compare two individuals that are both evaluated by simulation models. The population in Noisy-AMGA, however, has most individuals evaluated by the meta-models instead of numerical models. The selection test

must be adapted to incorporate the comparison between two meta-model-evaluated results and between meta-model-evaluated and numerical-model-evaluated results. Recall that in statistical decision theory the target function of the meta-models is the true conditional expectation when a squared error loss function is used. If we neglect the meta-model's prediction error (note that AMGA also ignored meta-model's prediction error in its selection operator, but GAs tolerated the small error), then the comparison between a meta-model-evaluated solution and a numerical-model-evaluated solution can be accomplished by a one sample student test as follows:

$$T_{n+1} = \left(\frac{(\overline{X}_A - \mu_B)}{\sqrt{\frac{s_A^2}{n}}} \right) \tag{6.7}$$

where T is a student distribution with n-1 degrees of freedom and n is the sample numbers of the PDE evaluated solution.

Using a similar argument, if two meta-model-evaluated individuals are compared, the predicted expectations can be directly compared.

6.3 Case Studies

The Noisy-AMGA was tested with the two groundwater remediation case studies presented in Chapter 2. The test cases have similar formulations as the deterministic optimization formulations, but an additional reliability constraint is included in the

formulations. The Noisy-AMGA parameter setups and the structures of the meta-models for these case studies are presented in the following sub-sections.

6.3.1 Noisy-AMGA Setups for the Hypothetical Case

The hypothetical groundwater remediation case study was presented in Chapter 3 and was solved deterministically in previous chapters. When this case study is solved under uncertainty, only the effects of the hydraulic conductivity uncertainty will be considered. A total of 1000 hydraulic conductivity realizations were generated using geostatistical conditional simulations from a lognormal distribution, following the approach of Smalley et al. (2000). The hydraulic conductivity lognormal distribution has a mean of 6.18*(m/day)*, a variance of 0.28 and a correlation length or integer scale of 2.8*(m*). The 1000 realizations served as a realization pool for sampling whenever a trial design was selected for numerical model evaluation. A total of 5 realizations were sampled using the incremental LHS for each design that was selected for numerical model evaluations. The sampling size was determined using the approach proposed by Gopalakrishnan et al. (2003).

To apply the incremental LHS algorithm, the hydraulic conductivity realizations must be ranked properly according to their effectiveness on the trial designs so that the realizations can be divided on equal ranges on the posterior cumulative probability. In this research, the ranking was approximated using the first moment of the hydraulic head difference described by Singh and Minsker (2003), which characterizes the spatial difference of two realizations.

Noisy-AMGA uses an additional reliability constraint to control the desired robustness of the optimal solutions. The reliability of a trial design denotes the fraction of sampled realizations that satisfy all constraints, which can be computed by the unbiased sample estimator. The desired reliability level was set to 0.9 for this case study. The reliability constraint is formulated as,

$$reliability \approx \frac{n_a}{N_a} \geq 0.9 \tag{6.8}$$

where n_a represents the total number of the effective samples (evaluated and accumulated in the cache) that satisfy equations 3.2, 3.3 and 3.4. N_a represents the total number of effective samples.

The modified Noisy-GA fitness function for this case study then becomes,

$$\min\left\{E\left[C_{TOT} + w_1 Viol_{Risk} + w_2 Viol_{head}\right] + w_3 Viol_{reliability}\right\} \tag{6.9}$$

where $Viol_{reliability}$ is the reliability violation if the evaluated reliability is less than the desired reliability level and w_3 is the reliability penalty weight. Since reliability is the major controlling constraint in Noisy-AMGA, w_3 is set to a higher value (2000 in this case study) than the risk and head constraint penalties. The risk and head constraints are still included in the fitness function because they provide selection pressure in early generations, when almost all trial designs have virtually the same reliability of zero.

Three ANNs are used as meta-models in this study. The details of the three ANN structures are listed in Table 6.1. The first two ANNs are trained with noisy data and they produce expected outputs. The third ANN is a deterministic ANN which predicts the

reliabilities of given trial designs. The first model uses pumping well locations and pumping rates as inputs to predict expected hydraulic heads at the pumping wells. The first ANN has 9 inputs, 12 nodes in the hidden layer and 3 outputs. The pumping well locations, the pumping rates, and the expected hydraulic heads at those wells are then used to feed a second model, which predicts the maximum log risk expectation at all monitoring wells. The log function is used to avoid scaling problems associated with extremely small numbers. The second ANN has 12 inputs, which are the pumping wells locations, the pumping rates and the three expected hydraulic heads predicted by the first ANN. The second ANN has 14 nodes in the hidden layer and it produces the expected log risk. The third ANN has the same inputs and the same number of hidden neurons as the second ANN, but it is established to predict the reliabilities of trial designs.

Table 6.1. ANN structures of the hypothetical case

	ANN-1	ANN-2	ANN-3
Inputs	$x_1, x_2, x_3, y_1, y_2, y_3, q_1, q_2, q_3$	$x_1, x_2, x_3, y_1, y_2, y_3, q_1, q_2, q_3, h_1, h_2, h_3$	same as ANN-2
Hidden neurons	12	14	14
Outputs	$E(h_1), E(h_2), E(h_3)$	*E(log-risk)*	*Reliability*

where (x_i, y_i) is the coordinate of pumping well i, q_i is the pumping rate of well i; h_i is the hydraulic head of pumping well *i; log-risk* is the log of the total risk (*TR*) in equation 3.2; and *reliability* is from equation 6.8.

6.3.2 Noisy-AMGA Setups for the Umatilla Case

Noisy-AMGA was also evaluated on the Umatilla case study detailed in Chapter 3 and was solved by AMGA in previous chapter. To consider uncertainty in this case study,

the hydraulic conductivity realizations were generated using the pumping test results performed by the Army Corps of Engineers (ACE) and the approach of Singh (2003). The Army Corps of Engineers (ACE) divided the first alluvial layer into 17 hydraulic conductivity zones as shown in Figure 6.2. The ACE performed a series of pumping tests from 47 monitoring wells in 9 of the 17 zones to determine the hydraulic conductivity values. The hydraulic conductivity values and boundaries of the other 8 zones where pumping test data are unavailable were estimated by the ACE after calibrating the model to observed annual average groundwater elevations. These hydraulic conductivity values are the primary source of hydrogeologic uncertainty. The uncertain hydraulic conductivity values in each of the 9 zones were modeled by a triangular distribution, with the boundaries shown in Table 6.2 (Singh, 2003). The lower and upper values of the triangular distributions were set to the minimum and maximum values of the ACE specified bounds, and the apex values were set to by the existing model values. By sampling from the triangular distributions, a total of 100 hydraulic conductivity realizations were generated. Similarly as the hypothetical case, the 100 realizations were ranked using the first moment of the hydraulic head difference (Singh and Minsker 2003). Of the 100 realizations, 4 realizations with unrealistically low conductivity values were discarded. Therefore the size of the realization sampling pool is 96. More details on the uncertainty modeling are given by Singh and Minsker (2003) and Singh (2004).

Table 6.2. Conductivity ranges from pumping tests.
Only Zones with sampling results are shown.

Zones	Existing Model Conductivity (ft/day)	Max Cond (ft/day)	Min Cond(ft/day)
2	3014	36551	319
5	2466	2673	2466
7	4110	4110	1128
8	600	3445	600
9	1500	5870	974
11	3014	8484	1867
12	1000	6376	1000
13	4932	4932	2617
15	1918	7401	1733

The additional reliability constraint given in Equation 6.8 was also included in the Umatilla case to control the desired robustness of the optimal solutions. The modified Noisy-AMGA fitness function of the Umatilla case study then becomes,

$$\min\left\{E\left[COST + w_1 Viol_{QE} + w_2 Viol_{RDX} + w_3 Viol_{TNT} + w_4 Viol_{Q}\right] + w_5 Viol_{reliability}\right\} \qquad (6.10)$$

where $Viol_{reliability}$ is the reliability violation if the evaluated reliability is less than the desired reliability level and w_5 is the reliability penalty weight. As in the hypothetical

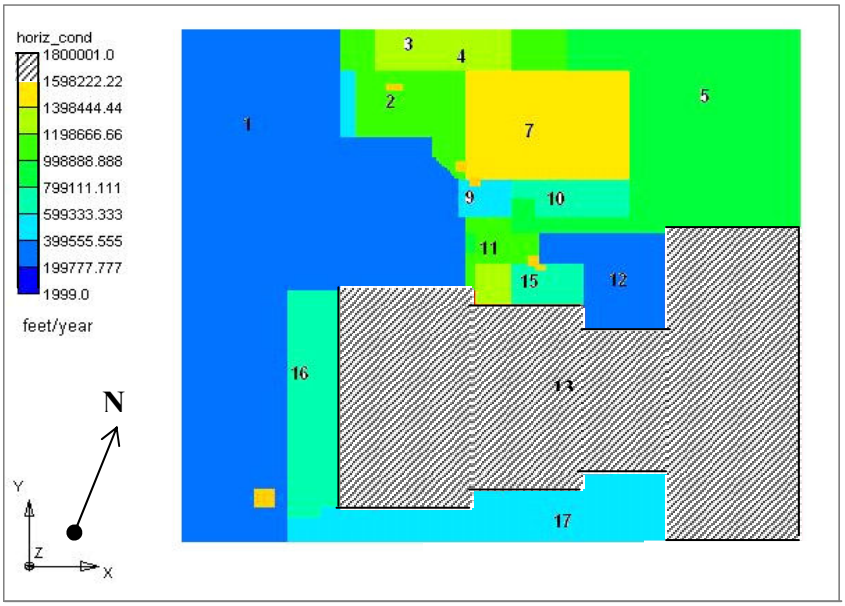

Figure 6.2. Hydraulic conductivity zones used in the Umatilla Model

case formulation, w_5 was set to a higher value (20000 in this case study) than the other penalties.

The total pumping rates of the deterministic solutions (Minsker et al. 2003, Chapter 4, Yan and Minsker 2006) are very close to the maximum allowable total pumping rate of 1170gpm. When uncertainty is considered, additional pumping is required that often violated the pumping constraint. Hence, the second formulation from Minsker et al (2003) was also used in addition to Formulation One, which increases the pumping capacity from 1170gpm to 1755gpm. The pumping rate constraint from Equation 3.7 then becomes,

$$\sum_{i \in z_1} q_i + \sum_{j \in z_2} q_j \leq 1755 \quad (gpm) \tag{6.11}$$

Three ANNs are also used as meta-models in this case study. Table 6.3 lists the detailed structures of the ANNs. The first ANN has 28 inputs, 12 hidden nodes, and three outputs. The second ANN has the same inputs, 14 hidden nodes, and two outputs. The first two ANNs are used to substitute for the flow and transport models. They have the same structures as the deterministic ANNs used in Chapter 4, but they are trained with noise data to produce expected outputs. These ANNs predict the expected value of a variable GAC cost component, and the expected values of the RDX and TNT maximum concentrations, which are then used to compute the expected violations caused by RDX and TNT plumes. The third ANN is a deterministic ANN, which has 28 inputs, 14 hidden nodes, and one output. The ANN is trained to predict the reliabilities of trial designs.

Table 6.3. ANN structures of the Umatilla case

	ANN-1	ANN-2	ANN-3
Inputs	$x_1, x_2, x_3, x_4, x_5, x_6, x_7, y_1, y_2, y_3, y_4, y_5, y_6, y_7, q_{1,nw}, q_{2,nw}, q_{3,nw}, q_{4,nw}, q_{5,nw}, q_{6,nw}, q_{7,nw}, q_{1,ow}, q_{2,ow}, q_{3,ow}, q_{4,ow}, q_{5,ow}, q_{6,ow}, q_{7,ow}$	same as ANN-1	same as ANN-2
Hidden neurons	12	14	14
Outputs	$E(C_{RDX(5)})$, $E(C_{TNT(5)})$, $E(VCG)$	$E(C_{RDX(5)})/ E(C_{RDX(4)})$, $E(C_{TNT(5)}) /E(C_{TNT(4)})$	*Reliability*

where (x_i, y_i) is the coordinate of new pumping well i; $q_{i,nw}$ is the pumping rate of new well i; $q_{i,ow}$ is the pumping rate of old well i; $C_{RDX(i)}$ and $C_{TNT(i)}$ are the maximum concentrations of RDX and TNT at the ith year; VCG is the variable GAC cost component from equation 3.6; and *reliability* is from equation 6.8

6.4 Results

This section presents results of applying Noisy-AMGA to the two test cases in the previous section. The Noisy-AMGA solutions are compared to the deterministic solutions to demonstrate how Noisy-AMGA achieved more robust solutions with more affordable computational requirements than a standard noisy GA.

6.4.1 Hypothetical Case Study Results

Noisy-AMGA was applied to the hypothetical case study using ten random initial populations (random seeds). Here we compare the solutions of Noisy-AMGA to solutions found by the deterministic AMGA using the same random seeds. The parameters controlling the PDE evaluation rate, initial training generations, sampling strategy and

ANN retraining frequency were determined in Chapter 4. The best parameter combination of, S_0=0.2, G_0=3, tournament+best sampling, and increased training set was used for the Noisy-AMGA runs. As in Chapter 4, the Noisy-AMGA runs were terminated after 96 generations. An offline analysis was then performed on both the AMGA and Noisy-AMGA solutions to analyze their robustness. Each solution was evaluated using all 1000 hydraulic conductivity realizations and the reliability was computed as the percentage of realizations that met all constraints.

Figure 6.3 compares detailed well locations and pumping rates for the solutions found in the first five AMGA and Noisy-AMGA runs. The other five runs converged to similar solutions and are not shown here for brevity. The figure shows that the pumping well locations for both AMGA and Noisy-AMGA solutions are similar. Both algorithms identified grid cells (6,5) and (7,4) as the two most likely cells for installing the pumping wells. Many other cell combinations nearby can produce almost the same objective function value as long as the wells are installed within this critical region so that the contaminants can be effectively intercepted. This is as expected since the hydraulic conductivity realizations were generated by a lognormal distribution with a relatively small variance (0.28). As a result, the pessimistic realizations are not much different from the optimistic realizations and the well deployment strategy is not significantly affected.

Figures 6.4 and 6.5 show the mean total costs and the evaluated reliabilities, respectively, for the 10 solutions found with AMGA and Noisy-AMGA. The error bars in the figure show the standard deviations of the total costs. The costs of the Noisy-AMGA

solutions ($188,500 on average) were slightly higher than those of AMGA solutions ($185,700 on average). The standard deviations of the total costs from the Noisy-AMGA and AMGA solutions were similar. However, the reliabilities of the Noisy-AMGA solutions were significantly better than the AMGA solutions. The average reliability of the Noisy-AMGA solutions was 0.97, which was more than a 100% improvement over the average reliability of the AMGA solutions, which was 0.47 on average. This confirmed that Noisy-AMGA was able to find more robust solutions with only slightly higher treatment cost than AMGA for this case study.

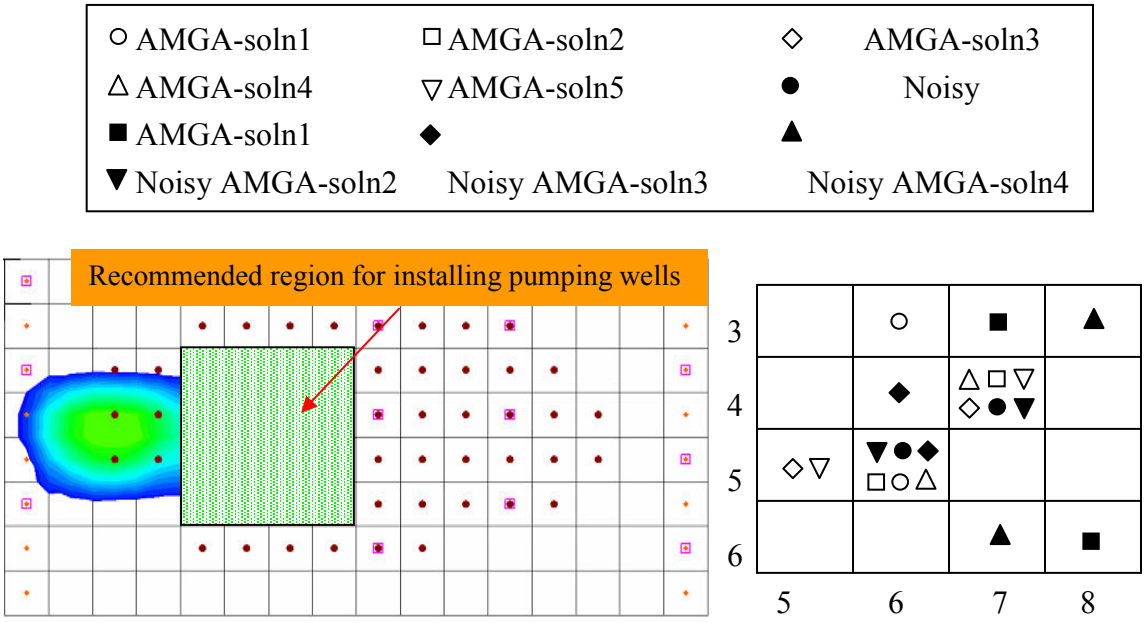

Figure 6.3. Detailed well locations for the first five AMGA and Noisy-AMGA solutions to the hypothetical case study. Numbers refer to the row and column numbers in the numerical model grid.

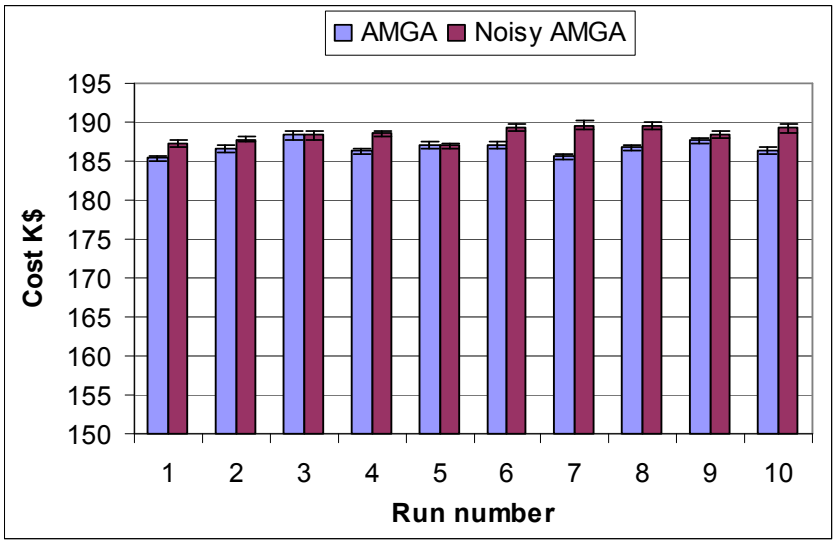

Figure 6.4. Costs of AMGA and Noisy-AMGA solutions to the hypothetical case study. The error bars show the standard deviations of the total costs.

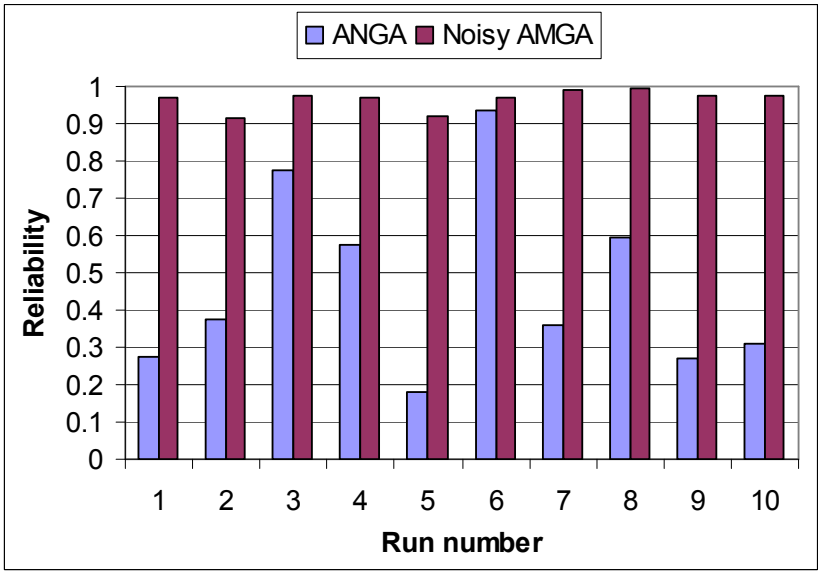

Figure 6.5. Reliabilities of AMGA and Noisy-AMGA solutions to the hypothetical case study.

Figure 6.6 shows the total pumping rates of the AMGA and Noisy-AMGA solutions. On average, the total pumping rates of the Noisy-AMGA solutions were 17% higher than those of the AMGA solutions. The increased pumping ensures that even if the

true hydraulic conductivity field is a pessimistic realization, the increased pumping capacity is still capable of cleaning up the contaminated site to the specified risk level. Since the two algorithms found similar pumping well locations, the redundant pumping capacity is the key factor for dealing with uncertainties in the hydraulic conductivities. The increased pumping rates also contributed to increasing the total cost.

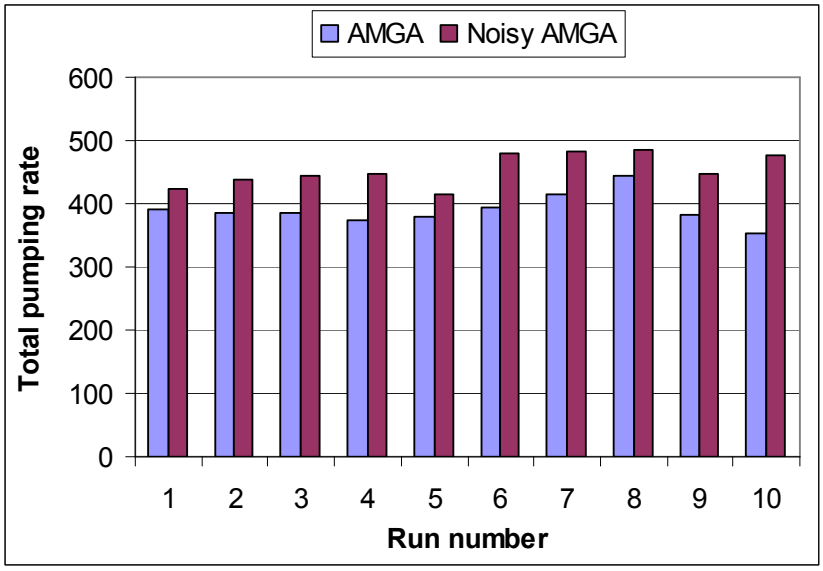

Figure 6.6. Total pumping rates for the AMGA and Noisy-AMGA solutions to the hypothetical case study.

Recall that Noisy-AMGA has an archiving cache (see 6.2.4) to store the history of sampling information and produce progressively improved fitness expectation estimations. Figure 6.7 shows how the cache increases the effective sample size as the optimization progresses. The population average of the effective sample size increases during the run due to the effect of the archive. The effective sample size of the best individuals oscillated many times before converging because the GA selected different individuals as the best ones over time. Many solutions were underestimated and appeared

to be good when they were sampled with only a few realizations, but after Noisy-AMGA resampled them in later generations they were discarded when more accurate averaging results were obtained. The sample size also affected the ANNs. In Noisy-AMGA runs the ANNs had poor performance in early generations because of the small sample number. The performance was improved as the sample number increased and the ANNs were adaptively retrained.

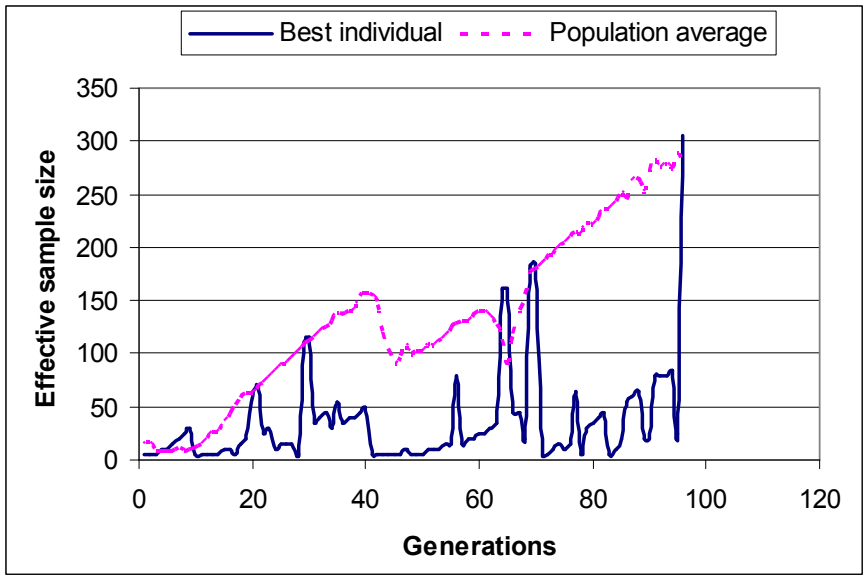

Figure 6.7. Effective sample size of the best individuals and the population average in a Noisy-AMGA run.

Finally, the Noisy-AMGA runs converged after about 4000 PDE evaluations. Recall that each PDE sampled trial design in the Noisy-AMGA framework was evaluated with 5 hydraulic conductivity realizations. Therefore the Noisy-AMGA runs required about 5 times more PDE evaluations than the AMGA runs. If a noisy-SGA with the same population size had been used, the total number of PDE evaluations would have been

48000. Therefore Noisy-AMGA saved more than 90% of PDE evaluations while achieving robust optimal solutions.

6.4.2 Umatilla Case Study Results

Noisy-AMGA was also applied to the Umatilla case study with ten random initial populations (random seeds) and compared with solutions found by AMGA using the same random seeds. The same AMGA parameters were used in the Noisy-AMGA runs as with the hypothetical case study. Similarly, an off-line test was performed in which each AMGA and Noisy-AMGA solution was evaluated using all 96 hydraulic conductivity realizations to compare reliabilities and costs.

Noisy-AMGA was first tested using formulation one from the ESTCP project. Figure 6.8 shows the achieved reliabilities of the Noisy-AMGA solutions. Although the Noisy-AMGA solutions had slightly better reliabilities than the AMGA solutions, the reliabilities were still poor and none of them achieved the target reliability (0.9) because of the strict constraint on the total pumping rate. This constraint was an active constraint in the deterministic AMGA, which used a relatively optimistic hydraulic conductivity realization. When a low conductivity realization is sampled by Noisy-AMGA, more pumping is required than is allowed by the strict total pumping constraint.

In order to achieve the desired reliability, Formulation Two was implemented next, which increases the maximum total pumping rate from 1175gpm to 1775gpm. With a deterministic optimization, Formulation Two resulted in the same solutions as Formulation One (Minsker et al., 2003).

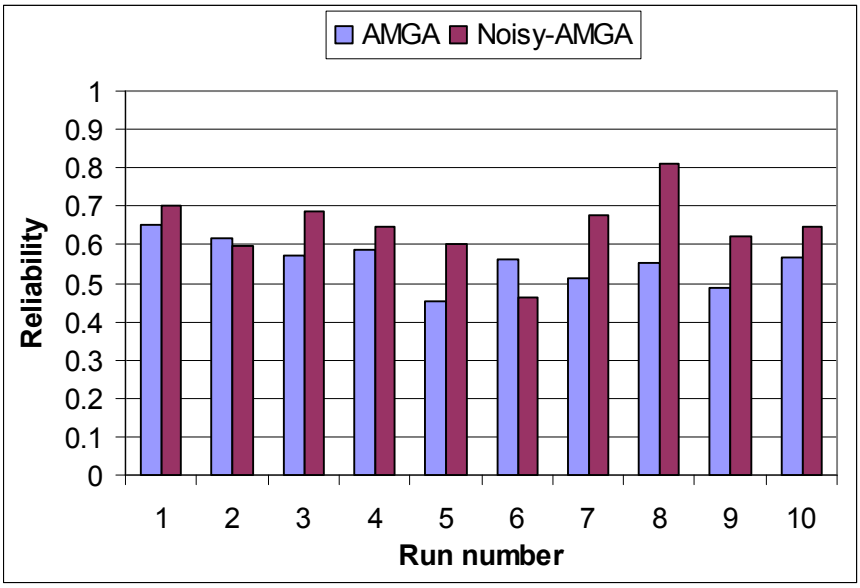

Figure 6.8. Reliabilities of AMGA and Noisy-AMGA solutions using Formulation One for the Umatilla case study

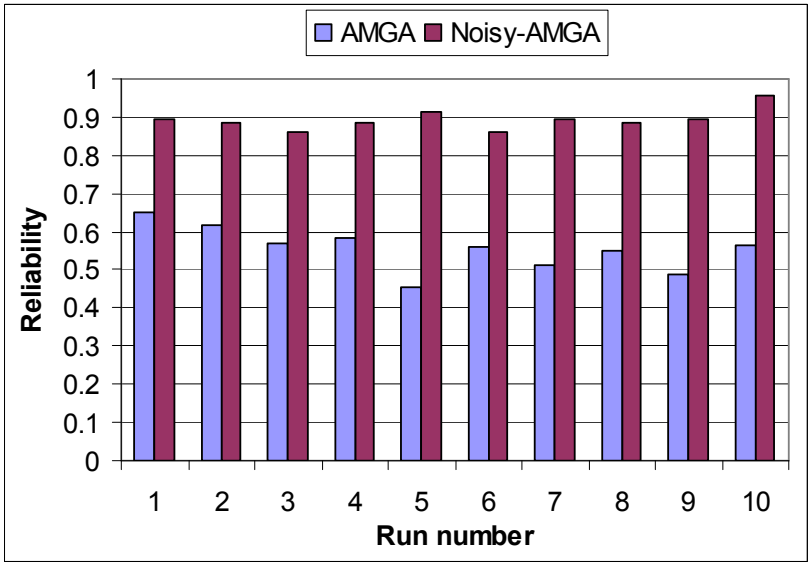

Figure 6.9. Reliabilities of AMGA and Noisy-AMGA solutions using Formulation Two for the Umatilla case study.

Figure 6.9 shows the evaluated reliabilities for the 10 solutions found with AMGA and Noisy-AMGA under Formulation Two. The reliabilities for the AMGA solutions were generally poor, ranging from 0.46 to 0.68. In contrast, the reliabilities for the Noisy-AMGA solutions ranged from 0.86 to 0.96. Those Noisy-AMGA solutions

with reliabilities slightly less than the desired reliability level (0.9) were caused by finite sampling levels. On average, the reliability of the Noisy-AMGA solutions was 0.89, which was a 55% improvement over the AMGA solutions (0.58).

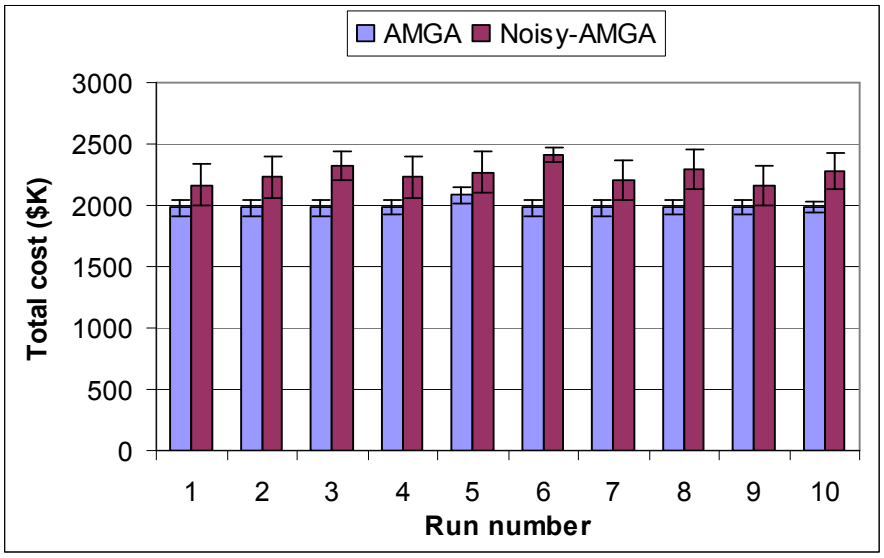

Figure 6.10. Mean total cost of AMGA and Noisy-AMGA solutions under Formulation Two for the Umatilla case study. The error bars show the standard deviations of the total costs.

Figure 6.10 shows the mean total costs for the ten AMGA and Noisy-AMGA solutions under Formulation Two. The error bars in the figure show the standard deviations of the total costs. To account for the uncertain hydraulic conductivity values, the optimal total pumping rates of the Noisy-AMGA solutions increased by 30% on average, which resulted in installation of a new GAC unit because the total pumping rates exceeded the capacity of the original treatment plant. The new GAC unit and the increased electricity cost contributed to the increased total cost. In terms of the cost standard deviations, the AMGA solutions are relatively low. This is because the AMGA solutions required a five-year cleanup time for most hydraulic conductivity realizations,

which makes the evaluated total costs cluster at the five-year cost range. In contrast, the increased pumping rates of the Noisy-AMGA solutions enabled more four-year cleanup realizations than the AMGA solutions. Therefore the evaluated costs of the Noisy-AMGA solutions have higher standard deviations, with both four-year (low) and five-year (high) solutions. On average, Noisy-AMGA optimal solutions involve a 13% increase in the expected total cost.

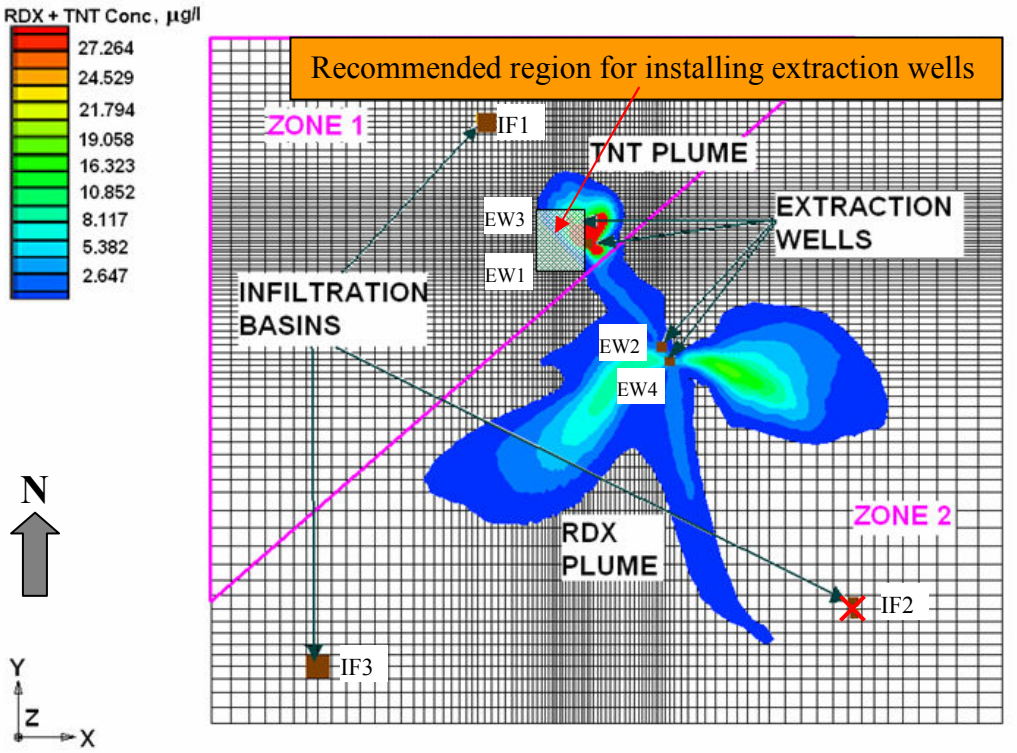

Figure 6.11. Noisy-AMGA solution overview of the Umatilla case study.

Figure 6.11 shows the overview of pumping well locations for the first five of Noisy-AMGA solutions (the other 5 solutions are similar). The Umatilla case hydraulic realizations were generated with triangular distributions that varied in a much larger range than the hypothetical case. As a result, the pumping well deployment strategy was

more significantly affected by the sampled realizations. Recall that the deterministic optimal solutions found by AMGA required the operation of the two recharge basins located in the south (IF2 and IF3) so that the RDX plume could be flushed toward the extraction wells located in the TNT plume. The more reliable solutions, however, did not use IF2, but instead used IF1 to flush the TNT plume toward the extraction wells. This modification occurs because the north-west corner of the TNT plume is located in a low hydraulic conductivity zone (zone 1, see Figure 6.2) that may be realized with very low conductivity values. The remaining TNT is difficult to remove in the low permeable zone but the recharge basin assists by flushing the remaining TNT into contiguous high permeable zones. Since IF2 is no longer used to flush the RDX plume in this solution, the extraction wells in the RDX plume (EW2 and/or EW4, which are not utilized in the deterministic solutions) are then needed to remove the RDX plume. The robust solutions also required installing new extraction wells in the TNT plume as detailed in Figure 6.12. The figure shows that the AMGA and Noisy-AMGA solutions share the same recommended region for installing the new extraction wells, but Noisy-AMGA required two or three additional wells in the region. When three wells are installed, only one of EW2 and EW4 is required; otherwise both are utilized. The two-well or three-well deployments are not significantly different in terms of the total cost, since the lower cost solutions (run 1 and run 9) used the two-well and three-well deployments, respectively.

The robust solutions indicate that removing the strongly sorptive TNT plume requires shifting the recharge basin toward the TNT plume and increasing the extraction

capacity. The underlying simulation model, which assumed equilibrium sorption, predicted that this deployment strategy could achieve the desired reliability. This assumption may not reflect the true geological condition, particular in the source region containing high TNT concentrations. Hence the reliabilities may be over estimated and further increases in pumping rates or additional wells may be necessary in practice.

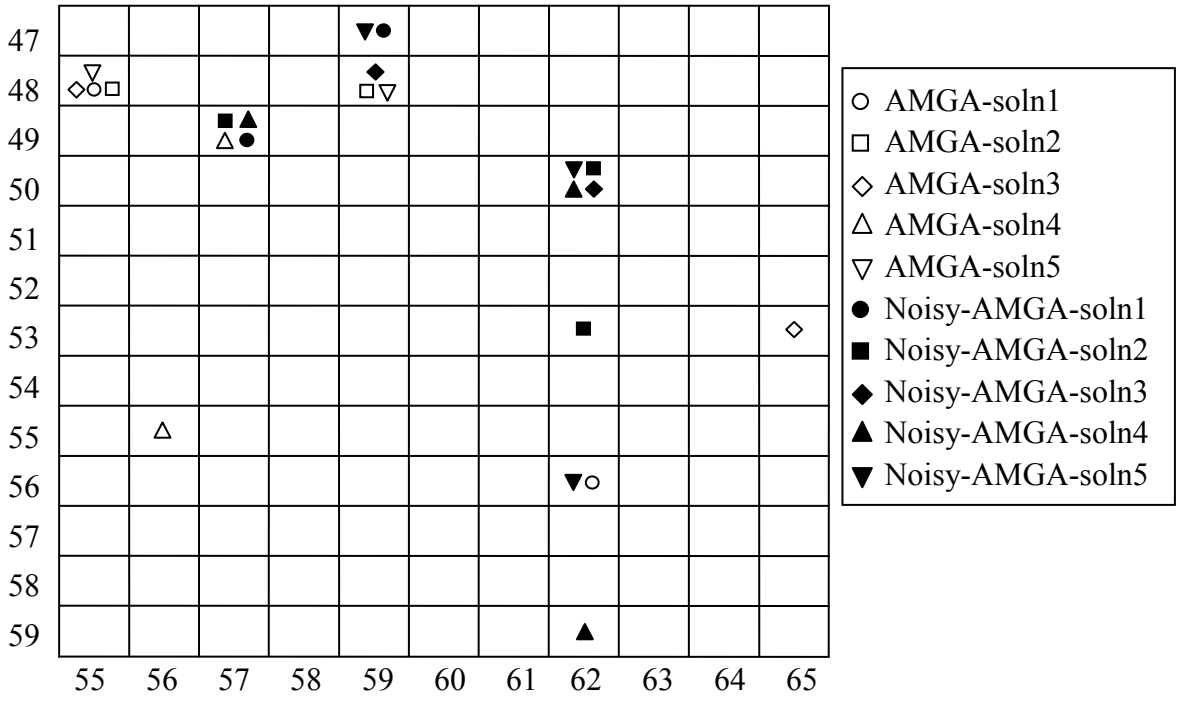

Figure 6.12. Well locations of the first five AMGA and Noisy-AMGA solutions for the Umatilla case study. The grid corresponds to the highlighted area in Figure 6.11.

The effective sample size for fitness averaging of a Noisy-AMGA run is shown in Figure 6.13. Compared to the hypothetical case, the Umatilla case has a much larger search space and the near-optimal solutions are located in a relatively small region. Figure 6.13 shows that the Umatilla case has more generations with low effective sample size than the hypothetical case (Figure 6.7). This confirms that the GA tested more trial designs before the optimal solution was identified and successfully reproduced. After

about 30 generations, the optimal solutions began to be resampled multiple times, and the high effective sample sizes ensured that the estimated reliabilities and fitness expectations were reasonably accurate for identifying good solutions.

Finally, Noisy-AMGA required on average 5,200 simulation evaluations for the Umatilla case. The AMGA runs converged after about 60~70 generations, which is less than the AMGA required generations (80~90). Compared to the AMGA runs, the Noisy-AMGA runs required only about 4.5 times more PDE evaluations although the incremental LHS sample size was 6. If a Noisy-SGA with no ANN and cache had been used to solve this problem, the total number of simulation evaluations would have been 37,800. Therefore Noisy-AMGA saved more than 86% of the simulation evaluations while achieving robust optimal solutions.

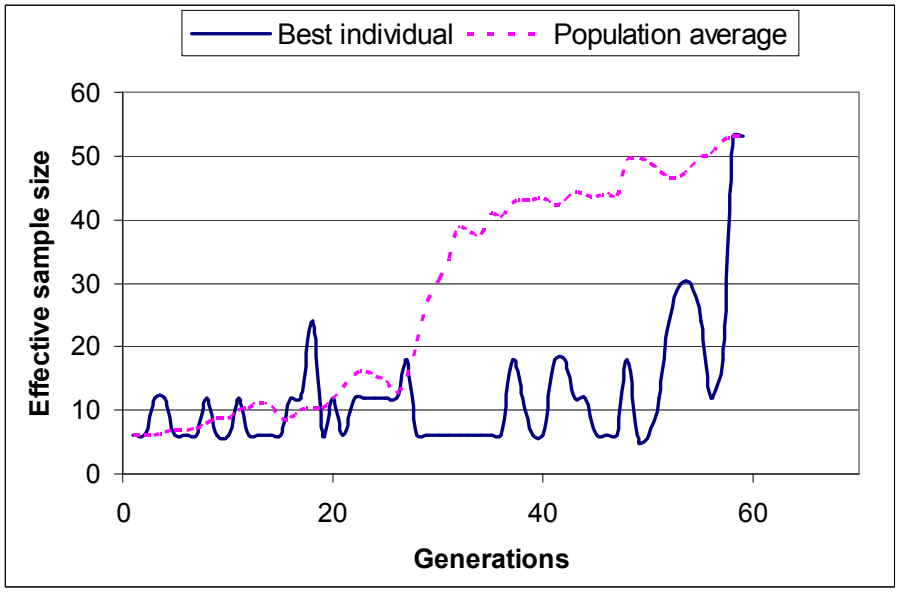

Figure 6.13. Effective sample size of the population average and the best individuals in a Noisy-AMGA run for the Umatilla case study.

6.5 Conclusions

Cleaning up contaminated groundwater sites is a costly but important task in environmental protection. Although state-of-the-art deterministic optimization approaches have been proposed to tackle the problems, the deterministic approaches can not tolerate parameter noise or uncertainty and may produce over-optimistic solutions with poor reliability. This chapter proposed a computationally efficient noisy genetic algorithm that utilizes the dynamic meta-model framework presented in Chapter 4. The algorithm, called Noisy-AMGA, combines a noisy GA with adaptively retrained artificial neural networks to produce robust solutions with substantial savings on computational overhead. In Noisy-AMGA, the ANNs are trained with noisy training data by a squared error loss function to predict the expectations of noisy outputs, which in turn are used to compute the fitness expectations. As with AMGA, the ANNs are periodically updated by injecting additional local solutions into the training sets. This technique improves the performance of ANNs in local as the GA converges.

Noisy-AMGA also implements an incremental Latin Hypercube sampling of hydraulic conductivity realizations to produce less biased expectation estimations with a small sample size. The sampled trial designs are stored into an archiving cache so that the effective sample size was progressively improved without incurring excessive samples in each generation. In addition, Noisy-AMGA modified the tournament selection operator. Depending on the comparison context, Noisy-AMGA uses different statistical tests to enable more realistic comparisons in a noisy optimization environment.

Using ANNs as approximation meta-models, Noisy-AMGA was tested on the hypothetical remediation case study and the Umatilla case study with multiple random initial populations. The Noisy-AMGA results were compared to the AMGA results performed on the same initial populations. Compared to the hypothetical case, the Umatilla case is a much harder case because the Umatilla hydraulic conductivities vary in much larger range. The total pumping rate must be relaxed to enable more pumping capacity. The results from both cases showed that highly reliable solutions could be found over all random seeds. On average, the Noisy-AMGA solutions improved the reliability by about 100% (hypothetical case) and 55% (Umatilla case) over the deterministic solutions. The reliable solutions had similar pumping well locations to the deterministic solutions in the hypothetical case, but different injection deployment and additional extraction wells were required in the Umatilla solutions. This also reveals that the pessimistic hydraulic conductivity realizations of the Umatilla case are noticeably different from the standard realization used in the deterministic optimization. In both cases, the total pumping rates were increased to tolerate possible variations in hydraulic conductivities. In terms of computational efficiency, the Noisy-AMGA saved 90% numerical model evaluations (hypothetical case) and 86% numerical model evaluations (Umatilla case) comparing to a noisy GA without ANN meta-models.

CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH

Genetic Algorithms are effective and robust optimization algorithms that have been used to optimize numerous complex water resources management problems. Understanding the underlying physical processes sufficiently for management often requires numerous computationally-intensive simulation model runs to identify optimal solutions, which poses a major computational barrier to effective use of GAs for complex problems. In this thesis, a dynamic Genetic Algorithm optimization framework was proposed that adaptively retrains surrogates to reduce computational effort without sacrificing performance. Techniques that can effectively coordinate the surrogates, the numerical models, and the optimization algorithms in the dynamic framework were also presented, and the proposed methods were tested on groundwater remediation case studies under both deterministic and noisy conditions. The obtained results indicate that the methods should be tested on other water resources management problems and optimization problems in other disciplines as well.

In this research a deterministic optimization approach, a dynamic meta-model genetic algorithm (AMGA) was first proposed. AMGA incorporates efficient surrogates (artificial neural networks) that are automatically trained and retrained to replace the time-consuming water resources simulation models. The deterministic version of AMGA was first applied to a hypothetical remediation case study with multiple random initial populations to identify the best parameter combination for AMGA. Using the best

parameter combination, AMGA was able to attain the desired optimal solutions with a savings of 90% of the simulation model evaluations. The best parameter combination was then used to apply AMGA to the Umatilla case study. Although the Umatilla case study had a much larger decision space, AMGA still successfully identified optimal solutions similar to solutions reported in previous SGA runs with a 86% reduction in the number of numerical model evaluations.

Results from these first applications of AMGA indicated that the performance of the embedded meta-models is a key factor in determining the quality of the achievable optimal solutions. Hence improving the meta-models' prediction accuracy in a dynamic environment is the second research objective, for which a hierarchical meta-model structure based on the divide-and-conquer concept was proposed. This hierarchical structure was implemented in a trust region-based adaptive meta-model GA (TRAMGA) and was then applied to the Umatilla case study using both ANNs and SVMs. The results showed that the TRAMGA runs converged to somewhat better optimal solutions and identified these solutions in fewer generations, regardless of which prediction tool was used. Moreover, the fitness prediction MSEs in the TRAMGA runs were smaller than in the AMGA runs. The improvement of TRAMGA over AMGA was greater when the ANNs were used as the prediction tools than the SVMs.

The hierarchical meta-model structure was also further extended and applied to predicting nitrate concentrations using the Sangamon case study published by Suen and Eheart (2003). In the case study a clustering algorithm was used to identify two local

clusters in the combined input and output space, and two local models were constructed in the cluster regions. The test results showed that the hierarchical meta-model structure was able to improve the nitrate predictions with both ANNs and SVMs. The trust region-based ANN achieved the best overall performance among all the tested prediction models.

The last section of this thesis extended AMGA to a stochastic GA optimization algorithm so that robust solutions can be identified to water resources problems under uncertainty. The stochastic optimization algorithm, called Noisy-AMGA, minimizes the expected objective function with a reliability constraint to achieve robust optimal solutions within a desired reliability level. In this algorithm the meta-models learn to predict the expected outputs using the squared error loss function and statistical t-tests are used in the GA's selection operator.

Applying Noisy-AMGA to the hypothetical case showed that the reliability (as measured by off-line simulations) of Noisy-AMGA solutions improved by about 100% over the deterministic solutions. The Noisy-AMGA solutions had similar pumping wells installed but the total pumping rates were increased to improve reliability. The results also showed that Noisy-AMGA saved more than 90% of the numerical model evaluations compared to a simple noisy GA. Similar results were obtained when Noisy-AMGA was applied to the Umatilla case study, where Noisy-AMGA improved reliabilities by 55% compared to the deterministic solutions. The robust Noisy-AMGA solutions required more pumping and a different recharge strategy to improve reliability. The required

numerical model evaluations in a Noisy-AMGA run was 86% less than the required evaluations in a noisy GA run.

The results of this research show considerable promise for adaptive meta-models to improve water resources optimization. A couple of future topics to extend this work are recommended. First, AMGA has an underlying distributed computing infrastructure to enable parallel computing of the simulation models. In each generation, AMGA uses sampling techniques to select a few valuable individuals to evaluate their fitness values by simulation models. This indicates that AMGA has to wait for at least one simulation model execution time in each generation no matter how many computing nodes are available. In later generations, this is not efficient because only a few computing nodes are utilized while the sampling rate is low. One approach that can improve the time efficiency is to use a generation based sampling strategy. In this approach, the sampling algorithm periodically samples a whole population and evaluates the individuals all at once. During the sampling intervals, the populations are all evaluated by the surrogates to avoid waiting time. This technique does not decrease the overall simulation model evaluations but it can reduce the computational time when a large number of computing nodes are available.

A second future research topic would be to improve the chromosome cache, which compares chromosomes bit by bit to determine if a chromosome is already stored in the memory structure. The caching efficiency can be improved if the chromosomes are compared by their decision variables instead, with a small error tolerance to enable soft

comparison (solutions are treated as equivalent if they are close enough to each other). Ideally, the tolerance should be updated from a larger value to a smaller value as the population converges. If the tolerance is appropriately controlled, the cache can save more PDE-evaluations with minimal loss in accuracy.

The third future topic would be to improve the hierarchical meta-model structure. In this research, the global model and the local models used the same prediction tool once it was chosen. Future research can explore techniques for combining heterogeneous prediction tools in a hierarchical structure. Note that different prediction tools are designed for different prediction purposes, and they usually have different performances on different training data sets. As a result, adopting different prediction tools according to their applicability has the potential to improve model predictions for complex systems. Likewise, the hierarchical modeling structure can be extended using decision trees (Russel and Norvig, 1995) to divide the modeling space into subspaces. Similar to the decision tree-based local linear models, local prediction models can be established on the subspaces maintained in the leaf nodes of decision trees.

The fourth future research topic would be to improve the performance of Noisy-AMGA through two approaches. The first approach would involve using critical realizations to reduce realization samples. Critical solutions are usually pessimistic realizations that are more difficult to satisfy. Critical realizations can be collected using techniques described by Ranjithan and Eheart (1993). The critical realizations are usually only a fraction of the total realizations, so the Latin Hypercube sample size can be further

reduced by creating a realization pool consisting of only critical realizations. The second approach is to provide good initial solutions to the Noisy-AMGA runs. The more robust solutions found in this research generally require more pumping to tolerate parameter variations, but the pumping well deployments are similar to the deployments of the deterministic solutions. The results indicate that the robustness of deterministic solutions may be substantially improved simply by slightly perturbing their decision variables. This observation implies that the deterministic solutions may carry valuable information on where to find more robust optimal solutions. Therefore Noisy-AMGA could use any available pre-existing deterministic solutions to generate initial populations with high probabilities assigned to trial designs close to the deterministic solutions. Alternatively the randomly perturbed initial solutions could be injected into Noisy-AMGA populations during the optimization processes.

The fifth future research topic would be to improve the Noisy-AMGA's objective function or the simulation models. In this research Noisy-AMGA minimizes the expected cost associated with remediation designs. The expected cost, however, is not necessarily the best approach for quantifying the balance between the gain and the risk for decision makers for real world engineering problems. The utility-of-money approach (ReVelle et al. 2004) could be used as a better fitness surrogate than the expected cost. The Noisy-AMGA can be modified to minimize the expected utility cost with a chosen utility function. Test runs are required to identify how the optimal design solutions are affected by using a utility function. In addition, the simulation models can also be improved. This

research assumed equilibrium sorption. This assumption may not be realistic (Luthy et al, 1997, Werth and Reinhard 1997a and 1997b), which would result in decreased reliabilities in the achieved solutions. More realistic simulation models should be tested in the future to determine the effectiveness of Noisy-AMGA.

The sixth future research topic would be to quantify the meta-models' prediction errors. In this research the meta-models' prediction errors were ignored, which makes the fitness comparisons fuzzy at times. Some good individuals may not be retained if high prediction errors are associated with them. A future research direction would be to explore efficient algorithms that require affordable iterations or simulations to quantify the prediction errors. If sufficient computing nodes are available in the distributed computing infrastructure, less efficient algorithms such as bootstrapping could also be used to quantify the error range within a desired computational time constraint.

Finally, this research focused on solving optimization problems with a single objective, but the dynamic surrogate modeling techniques should be extended in future work to solve multi-objective optimization problems. For example, Singh and Minsker (2004) used a noisy non-dominated sorted GA (NSGA) to optimize groundwater remediation designs, the noisy-NSGA can replace the standard GA or standard Noisy-GA framework with little modifications of the meta-model modeling techniques, but effort must be taken to identify the appropriate meta-modeling setups that can work efficiently with multiple-objective GAs.

REFERENCES

Alexandrov, N. M., J. E. Dennis Jr., R. M. Lewis, and V. Torczon (1998), A Trust-region Framework for Managing the Use of Approximation Models in Optimization, *Struc. Optimiz.*, *15*(1), 16-23.

Aly, A. H., and R. C. Peralta (1999), Optimal design of aquifer cleanup systems under uncertainty using a neural network and a genetic algorithm, *Water Resour. Res., 35*(8), 2523-2532.

Ahfeld, D. P., J. M. Mulvey, G. F. Pinder, and E. F. Wood (1988), Contaminated groundwater remediation design using simulation, optimization, and sensitivity theory. 1. Model development, *Water Resour. Res., 24*(3), 431-441.

Arnold, J. G., J. R. Williams, R. Srinivasan, and K. W. King (1996), SWAT—Soil and water assessment tool: Users manual, U.S. Department of Agriculture-ARS, Temple, Tex.

Asefal, T., M. Kemblowski1, G. Urroz, and M. McKee1 (2005a), Support vector machines (SVMs) for monitoring network design, *Ground Water*, 43(3), 413.

Asefal, T., M. Kemblowski1, G. Urroz1, U. Lall, and G. Urroz (2005b), Support vector machines for nonlinear state space reconstruction: Application to the Great Salt Lake time series, *Water Resour. Res.*, *41*(12).

Bear, J., and Y. Sun (1998), Optimization of pump-treat-inject (PTI) design for the remediation of a contaminated aquifer: multi-stage design with chance-constraints, *Journal of Contaminant Hydrology*, *29*(3), 223-242.

Balla, M. C. and S. Lingireddy (2000), Distributed Genetic Algorithm Model on a Network of Personal Computers, *J. Computing in Civil Engineering*, *14*(3), 199-205.

Bau, D. and A. S. Mayer (2006), Stochastic Management of Pump-and-Treat Strategies Using Surrogate Functions, Accepted for publication in Adv. Water Resources.

Brooker , A. J., J. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. Trosset (1998), A rigorous framework for optimization of expensive functions by surrogates, *Struc. Optimiz.*, *17*(1), 1–13.

Chan, N. (1993), Robustness of the multiple realization method for stochastic hydraulic aquifer management, *Water Resour. Res.*, *29* (9), 3159-3167.

Chan, N. (1994), Partial infeasibility method for chance-constrained aquifer management, *J. Water Resour. Plann. Manage.*, *120*(1), 70-89.

Chan Hilton, A. B. and T. B. Culver (2005), Groundwater remediation design under uncertainty using a robust genetic algorithm, *J. Water Resour. Plann. Manage.*, *131* (1), 25-34.

Chan Hilton, A. B. and Y. Li (2005), Optimal groundwater sampling network design through ant colony optimization, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)* , June 25-29, Washington, DC.

Chang, C.-C. and C.-J. Lin (2001), LIBSVM: a library for support vector machines, Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chang, L-C., C. A. Shoemaker, and P. L-F. Liu (1992), Optimal time varying pumping rates for groundwater remediation: Application of a constrained optimal control theory, *Water Resour. Res.*, *28*(12), 3157-3574.

Charnes, A. and W. W. Cooper (1959), Chance-constrained programming, *Management Science*, 6(1)

Cooper, G., R. C. Peralta, and J. J. Kaluarachchi (1998), Optimizing separate phase light hydrocarbon recovery from contaminated unconfined aquifers, *Adv. Water Resour.*, *21*(5), 339-350.

Coppola Jr., E., F. Szidarovszky, M. Poulton and E. Charles (2003), An Artificial Neural Network Approach for Predicting Transient Water Levels in a Multilayered Groundwater System Under Variable State, Pumping, and Climate Conditions, *J. Hydrol. Eng.*, *8*(6), 348-360.

Clement, T. P (1997), RT3D - A modular computer code for simulating reactive multi-species transport in 3-dimensional groundwater aquifers, Battelle Pacific Northwest National Laboratory Research Report, PNNL-SA-28967.

Clement, T. P., C. D. Johnson, Y. Sun, G. M. Klecka, and C Bartlett (2000), Natural attenuation of chlorinated solvent compounds: model development and field-scale application, *J. Contam. Hydrol., 42*, 113-140.

Clement, T. P, Y. Sun, B. S. Hooker, and J. N Petersen (1998), Modeling multi-species reactive transport in groundwater, *Ground Water Monitoring and Remediation*, *18*(2), 79-92.

Culver, T. B., and C. A. Shoemaker (1992), Dynamic optimal control for groundwater remediation with flexible management periods, *Water Resour. Res., 28*(3), 629-641.

Culver, T. B., and C. A. Shoemaker (1993), Optimal control for groundwater remediation by differential dynamic programming with quasi-Newton approximations, *Water Resour. Res.*, *29*(4), 823-831.

Culver, T. B., and C. A. Shoemaker (1997), Dynamic optimal groundwater reclamation with treatment capital costs, *J. Water Resour. Plann. Manage.*, *123*(1), 23-29.

Dibike Y. B. and D. Solomatine (2001), River Flow Forecasting Using Artificial Neural Networks, *Journal of Physics and Chemistry of the Earth*, Part B: Hydrology, Oceans and Atmosphere, *26*(1), 1-8.

Dougherty, D. E., and R. A. Marryott (1991), Optimal groundwater management 1. Simulated Annealing, *Water Resour. Res.*, *27*(10), 2493-2508.

Eheart, J. W., S. E. Cieniawski, and S. Ranjithan (1993), Genetic-algorithm-based design of groundwater quality monitoring system, WRC Research Report No. 218. Urbana: Department of Civil Engineering, The University of Illinois at Urbana-Champaign.

Fitzpatrick J. M and J. J. Grefenstette (1988), Genetic algorithms in noisy environments, *Machine Learning*, *3*, 101-120.

Freyberg, D. L., (1986), A natural gradient experiment on solute transport on a sand aquifer, 2, Spatial movements and the advection and dispersion of nonreactive tracers, *Water Resour. Res.*, *22*(13), 2031-2046.

Gailey, R. M., and S. M. Gorelick (1993), Design of optimal, reliable plume capture schemes: Application to the Gloucester landfill ground-water contamination problem, *Ground Water*, *31*(1), 107-114.

Garrett Jr., J. H., S. Ranjithan, and J. W. Eheart (1992), Application of Neural Networks to Groundwater Remediation, a chapter in ASCE Monograph: Expert Systems in Civil Engineering - Knowledge Representation, ed. R. Allen, ASCE, New York.

Goldberg, D. E. (1989), *Genetic algorithms in search, optimizations and machine learning*, Addison –Wesley, New York, NY.

Gopalakrishnan, G., B. S. Minsker, and D. Goldberg (2003), Optimal sampling in a noisy genetic algorithm for risk-based remediation design, *Journal of Hydroinformatics*., *5*, 11-25

Gorelick, S. M., C. I. Voss, P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright (1984), Aquifer reclamation design: The use of contaminant transport simulation combined with nonlinear programming, *Water Resour. Res.*, *20*(4), 415-427.

Hagan, M. T. and M. Menhaj (1994), Training feedforward networks with the marquardt algorithm, *IEEE Transactions on Neural Networks*, *5*(6), 989-993.

Hastie, T., R. Tibshirani and J. H. Friedman (2001), *The Elements of Statistical Learning : data mining, inference, and prediction*, New York : Springer.

Holland, J. H. (1975), *Adaptation in natural and artificial system*, University of Michigan Press, Ann Arbor, Michigan.

Hughes, E. J. (2001), Evolutionary Multi-Objective Ranking with Uncertainty and Noise, Proceedings of the First International Conference Evolutionary Multi-Criterion Optimization, Zurich, Switzerland, March 7-9, 329-343.

Jacobs, R. A. and M. I. Jordan (1993), Learning piecewise control strategies in a modular neural network architecture, *IEEE Trans. Syst. Man Cybern.*, *23*(2) 337-345.

Jin, Yaochu, M. Olhofer, and B. Sendhoff (2002), A Framework for Evolutionary Optimization With Approximate Fitness Functions, *IEEE Transactions on Evolutionary Computation, 6*(5), 481-494.

Jordan, M. I., and R. A. Jacobs (1994), Hierarchical mixture of experts and the EM algorithm, *Neural Comput.*, *6*, 181-214.

Karatzas, G. P., and G. F. Pinder (1993), Groundwater management using numerical simulation and outer approximation method for global optimization, *Water Resour. Res.*, *29*(10), 3371-3378.

Karatzas, G. P., and G. F. Pinder (1996), The solution of groundwater quality management problems with a nonconvex feasible region a cutting plane optimization technique, *Water Resour. Res.*, *32*(4), 1091-1100.

Kaufman L. and P. J. Rousseeuw (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley.

Khan, M. S. and P. Coulibaly (2006), Predicting monthly Lake water levels using support vector machine, *ASCE Journal of Hydrologic Engineering* (in press, accepted 05-16-05).

Kratica , J. (1999), Improving performances of the genetic algorithm by caching, *Computers and Artificial Intelligence*, *18*(3), 271—283.

Lefkoff, L. J., and S. M. Gorelick (1986), Design and cost analysis of rapid aquifer restoration systems using flow simulation and quadratic programming, *Ground Water*, *24*(6), 777-790.

Lewis, H. R. and L. Denenberg (1997), *Data Structures and Their Algorithms*, Addison Wesley, New York, NY.

Liong, S. Y. and C. Sivapragasam (2002), Flood Stage Forecasting with Support Vector Machine, *Journal of American Water Resources Association*, *38*(1), 173-186.

Liu, Y. (2001), Multiscale approach to optimal control of in-situ bioremediation of groundwater, PhD dissertation, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign.

Liu, Y., B. S. Minsker, and F. Saied (2001), One-way spatial multiscale method for optimal bioremediation design, *J. Water Resour. Plann. Manage.*, *127*(2), 130-139.

Liu, Y., and B. S. Minsker (2002), Efficient multiscale methods for optimal in situ bioremediation design, *J. Water Resour. Plann. Manage.*, *128*(3), 227-236.

Luthy, R. G., G. R. Aiken, M. L. Brusseau, S. D. Cunningham, P.M. Gschwend, J. J. Pignatello, M. Reinhard, S. J. Traina, W. J. Weber Jr., and J. C. Westall (1997), Sequestration of Hydrophobic Organic Contaminants by Geosorbents, *Environ. Sci. Technol. & Technology*, *31*(12), 3341-3347.

Mansfield, C. M., C. A. Shoemaker, and L-Z. Liao (1998), Utilizing sparsity in time-varying optimal control of aquifer cleanup, *J. Water Resour. Plann. Manage.*, *124*(1), 39-46.

Maskey, S., Dibike, Y. B. Jonoski, and D. P. Solomatine (2000), Groundwater model approximation with artificial neural network for selecting optimal pumping strategy for plume removal, In: *AI Methods in Civil Engineering Applications* (O. Schleider, A. Zijderveld, eds), Cottbus, 67-80.

McDonald, M. G., and Harbaugh (1988), A.W. A modular three-dimensional finite-difference ground-water flow model. Techniques of Water Resources Investigations 06-A1, United States Geological Survey.

McKay, D. M, R. J. Bechman, and W. J. Conover (1979), A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code, *Technometrics*, *21*(2), 239-245.

McKay, D. M., D. L. Freyberg, and P. V. Roberts (1986), A natural gradient experiment on solute transport on a sand aquifer, 1, Approach and overview of plume movement, *Water Resour. Res.*, *22*(13), 2017-2029.

McKinney, D. C., and M. D. Lin (1994), Genetic algorithm solution of groundwater management models, *Water Resour. Res.*, *30*(6), 1897-1906.

Mckinney, D. C., and M. D. Lin (1995), Approximate mixed-integer nonlinear programming methods for optimal aquifer remediation design, *Water Resour. Res.*, *31*(3), 731-740.

Mckinney, D. C., and M. D. Lin (1996), Pump-and-treat groundwater remediation system optimization, *J. Water Resour. Plann. Manage.*, *122*(2), 128-136.

Miller, B. L., and D. E. Goldberg (1996), Optimal sampling for genetic algorithms, in Intelligent Engineering Systems Through Artificial Neural Networks, vol. 6, *Smart Engineering Systems: Neural Networks, Fuzzy Logic, and Evolutionary Programming*, edited by C. H. Dagli et al., 291-297, Am. Soc. of Mech. Eng., New York.

Minsker, B. S., and C. A. Shoemaker (1998a), Computational issues for optimal in-situ bioremediation design, *J. Water Resour. Plann. Manage.*, *124*(1), 39-46.

Minsker, B. S., and C. A. Shoemaker (1998b), Dynamic optimal control of in-situ bioremediation of ground water, *J. Water Resour. Plann. Manage.*, *124*(3), 149-161.

Minsker, B. S., Y. Zhang, R. Greenwald, R. Peralta, C. Zheng, K. Harre, D. Becker, L. Yeh, and Yager (2003), K, Final Technical Report for Application of Flow and Transport Optimization Codes to Groundwater Pump and Treat Systems, Environmental Security Technology Certification Program (ESTCP), Available at http://www.frtr.gov/estcp/

Morgan, D. R., J. W. Eheart, and A. J. Valocchi (1993), Aquifer remediation design under uncertainty using a new chance constrained programming technique, *Water Resour. Res.*, *29*(3), 551-562.

Neal, R. M. (1996), *Bayesian Learning for Neural Networks*, Springer-Verlag, New York.

National Research Council (1994), Alternatives for Ground Water Cleanup, National Academy Press, Washington, D.C.

Peralta, R. C. (2002), Optimal Pumping Strategies For Umatilla Chemical Depot RDX And TNT Plumes, Draft final report presented to Navy Facilities Engineering Command, Systems Simulation/Optimization, Utah State.

Pleming, J. B. and R. D. Manteufel (2005), Replicated Latin Hypercube Sampling, 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference, Austin, Texas.

Polak, E. (1971), *Computational Methods in Optimization*. New York: Academic Press.

Ranjithan, S, J. W. Eheart, and J. H. Garrett Jr. (1993), Neural net work-based screening for groundwater reclamation under uncertainty, *Water Resour. Res., 29*(3), 563-574.

Reed, P. M., B. S. Minsker, and D. E. Goldberg (2000), Designing a Competent Simple Genetic Algorithm for Search and Optimization, *Water Resour. Res.*, *36*(12), 3757-3761.

Reed, P., B. S. Minsker, and D. E. Goldberg (2001), A multiobjective approach to cost effective long-term groundwater monitoring using an Elitist Nondominated Sorted Genetic Algorithm with historical data, *Journal of Hydroinformatics*, *3*(2), 71-90.

Reed, P., B. S. Minsker, and D.E Goldberg (2003), Simplifying Multiobjective Optimization: An Automated Design Methodology for the Nondominated Sorted Genetic Algorithm-II, *Water Resour. Res.*, *39*(7).

Ren, X., and B. S. Minsker (2005), Which Groundwater Remediation Objective Is Better, A Realistic One Or A Simple One?, *J. Water Resour. Plann. Manage., 131*(5), 351-361.

ReVelle, C. S., E. E. Whitlatch Jr, and J. R. Wright (2004), *Civil and Environmental Systems Engineering*, Second Edition, Pearson Prentice Hall Inc, New Jersey.

Ritzel, B. J., J. W. Eheart, and S. Ranjithan (1994), Using genetic algorithms to solve a multiple-objective groundwater pollution containment problem, *Water Resour. Res.*, *30*(5), 1589-1603.

Rizzo, D. M., and D. E. Dougherty (1996), Design Optimization for Multiple Management Period Groundwater Remediation, *Water Resour. Res.*, *32*(8), 2549-2561.

Rogers, L. L., and F. U. Dowla (1994), Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling, *Water Resour. Res., 30*(2), 457-481.

Rogers, L. L., F. U. Dowla, and V. M. Johnson (1995), Optimal Field-Scale Groundwater Remediation Using Nerual Networks and the Genetic Algorithm, *Environ. Sci. Technol*., *29*(5), 1145-1155.

Russel, S., P. Norvig (1995), *Artificial Intelligence-A Modern Approach*, Prentice Hall, Englewood Cliffs, New Jersey.

Sawyer, C. S., Y. Lin (1998), Mixed-Integer Chance-Constrained Models for Ground-Water Remediation, *J. Water Resour. Plann. Manage*., *124*(5), 285-294.

Schölkopf, B. (1999), Support vector learning, Tutorial given at DAGM'99.

Singh, A. (2003), Uncertainty Based Multi-Objective Optimization Of Groundwater Remediation Design, M. S. Thesis, University of Illinois.

Singh, A., and B. S. Minsker (2003), Modeling and Characterization of Uncertainty for Optimization of Groundwater Remediation at the Umatilla Chemical Depot, ASCE. EWRI, 2003, Philadelphia, PA.

Singh, A., and B. S. Minsker (2004), Uncerntainty Based Multi-Objective Optimization of Groundwater Remediation at the Umatilla Chemical Depot, ASCE. EWRI. 2004, Salt Lake City, UT.

Smalley, J. Bryan, B. S. Minsker, and David E. Goldberg (2000), Risk-based in situ bioremediation design using genetic algorithm, *Water Resour. Res., 36*(10), 3043-3051.

Sudicky, E. A. (1986), A natural gradient experiment on solute transport in a sand aquifer: Spatial variability of hydraulic conductivity and its role in the dispersion process, *Water Resour. Res.*, *22*(13), 2069-2082.

Suen, J-P. and J. W. Eheart (2003), Evaluation of Neural Networks for Modeling Nitrate Concentrations in Rivers, *J. Water Resour. Plng. and Mgmt.*, *129*(6), 505-510.

Teich, J. (2001), Pareto-Front Exploration with Uncertain Objectives, Proceedings of the First International Conference Evolutionary Multi-Criterion Optimization, Zurich, Switzerland, March 7-9, 2001, 314-328

Tiedeman, C. and S. M. Gorelick (1993), Analysis of uncertainty in optimal groundwater contaminant capture design, *Water Resour. Res., 29*(7), 2139-2153.

U.S. Army Corps of Engineering (USACE) (1996), Final remedial design submittal, contaminated groundwater remediation, explosives washout lagoons, Umatilla Depot Activity, Hermiston, OR.

U.S. Army Corps of Engineering (USACE) (2000), Explosives washout lagoons groundwater model revision (preliminary draft), Umatilla Chemical Depot, Hermiston, OR.

U.S. Environmental Protection Agency (EPA) (1997), Cleaning Up the Nationa's Waste Sites: Markets and Technology Trends. EPA 542-R-96-005. Washington, D.C.: EPA, Office of Solid Waste and Emergency Response.

Vermeulen, P. T. M., A. W. Heemink and C. B. M. Testroet (2002), Reduction of large-scale numerical ground water flow models. In: S. Majid Hassanizadeh, R.J. Schotting, e.a. (eds.); Proceedings of the XIVth Int. Conference on Computational Methods in Water Resources, Delft, The Netherlands, ISBN 0-444-50975-5, 397-404.

Wagner, B. J. and S. M. Gorelick (1987), Optimal groundwater quality management under parameter uncertainty, *Water Resources Research*, *23* (7), 1162-1174.

Wagner, B.J. and S. M. Gorelick (1989), Reliable aquifer remediation in the presence of spatially variable hydraulic conductivity: from data to design, *Water Resour. Res.*, *25* (10), 2211–2225

Werth, C. J. and M. Reinhard (1997a), Effects of Temperature on Trichloroethylene Desorption from Silica Gel and Natural Sediments. 1. Isotherms, *Environ. Sci. Technol*., *31*(3), 689-696.

Werth, C. J. and M. Reinhard (1997b), Effects of Temperature on Trichloroethylene Desorption from Silica Gel and Natural Sediments. 2. Kinetics, *Environ. Sci. Technol*., *31*(3), 697-703.

Whiffen, G. J. (1995), Optimal control for deterministic and uncertain groundwater remediation. PhD dissertation, School of Civil and Environmental Engineering, Cornell Univ., Ithaca, New York.

Yan, S., and B. S. Minsker (2006), Optimal groundwater remediation design using an Adaptive Neural Network Genetic Algorithm, *Water Resour. Res.*, *42*.

Yong, J. H., and C. A. Shoemaker (1999), Comparison of optimization methods for ground-water bioremediation, *J. Water Resour. Plann. Manage.*, *125*(1), 54-63.

Zhang, B., and R. S Govindaraju (2000), Prediction of watershed runoff using Bayesian concepts and modular neural networks, *Water Resour. Res.*, *36*(3), 753–762

Zhang, B. T. (1994), An incremental learning algorithm that optimizes network size and sample size in one trial, *Proceedings of International Conference on Neural Networks*, 215-220. Orlando.

Zheng, C. and P. P. Wang (1996), Parameter structure identification using tabu search and simulated annealing*, Adv. Water Resour.,19*(4), 215-224.

Zheng, C. and P. P. Wang (1999), MT3DMS: Documentation and User's Guide, Report to the US Army Corps of Engineers Waterways Experiment Station, (available at http://hydro.geo.ua.edu).

Zheng, C. and P. P. Wang (2002), A Field Demonstration of the Simulation-Optimization Approach for Remediation System Design, *Ground Water, 40*(3), 258-265.

AUTHOR'S BIOGRAPHY

Shengquan Yan received his Bachelor of Engineering and Master of Engineering degrees, both in Environmental Engineering, from Tongji University, Shanghai, China, in 1996 and 1999. During his graduate study from 1996 to 1999 in Tongji University, he developed WSZ – a water distribution pipe-net simulation that is used by several water companies in China. From 1999 to 2000, he worked in Hongyang water modeling company as a team leader in designing software systems for modeling water distribution systems. One year later, he worked as a research assistant in National Institute for Agricultural Research, France to develop a web-based ecological knowledge information system. In August 2001, he was admitted to the Ph.D. program in Civil and Environmental Engineering at the University of Illinois at Urbana-Champaign. His research interests include optimal groundwater remediation design, statistical learning, modeling large and complex water resources systems, engineering risk analysis, and environmental information technology.