# ABSTRACT

MOHAMED SAYEED. *A parallel optimization framework for inverse problems.* (under the direction of Dr. G. Mahinthakumar)

Inverse problems that are constrained by large-scale partial differential equation (PDE) systems demand significant computational resources. These problems generally require the solution of a large number of complex PDE systems. Three dimensional subsurface characterization inverse problems fall under this category. A parallel hybrid optimization framework using global search and local search (LS) techniques is developed. The global search uses genetic algorithms (GAs). For LS several non-gradient based algorithms such as Nelder-Meade simplex, Hooke-Jeeves pattern search and Powell's method of conjugate directions and a gradient based algorithm namely, Fletcher-Reeves conjugate gradient method are implemented in the framework. Subsurface inverse characterization problems are posed as optimization problems and solved using this framework. The GA or hybrid GA-LS optimizer is employed to drive a parallel finite-element (FEM) groundwater transport simulator. Multilevel parallelism opportunities exist at the coarse-grained optimization level and the fine-grained function evaluation level. Coarse-grained parallelism (task parallelism) in the optimizer is exploited using a self-scheduling algorithm. Fine-grained parallelism (data parallelism) in the FEM transport simulator is achieved through a domain decomposition strategy. The MPI (Message Passing Interface) communication library is used for the parallel implementation. Parallelism is enhanced for local searches by enabling concurrent execution of multi-start or multi-type local searches. Performance results for convergence are examined for different test problems including biological activity zone identification,

contaminant source zone identification (location and concentration) and contaminant sources release history reconstruction problems showing the applicability of the proposed approach. The size and complexity of problems solved in this research far exceed what has been reported to date in the literature. The implementation has been extensively tested on a single supercomputer and on the grid (TeraGrid). This research illustrates that the hybrid approaches are generally more effective than either standalone GA or LS for solving inverse problems.

# AN EFFICIENT PARALLEL OPTIMIZATION FRAMEWORK FOR INVERSE PROBLEMS

by

**MOHAMED SAYEED**

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

**DEPARTMENT OF CIVIL, CONSTRUCTION AND ENVIRONMENTAL ENGINEERING**

Raleigh

2003

**APPROVED BY:**

| | |
|---|---|
| **(Dr. Ranji S. Ranjithan)** | **(Dr. Abhinav Gupta)** |
| Co-chair of Advisory Committee | Co-chair of Advisory Committee |
| **(Dr. G. Mahinthakumar)** | **(Dr. John. W. Baugh Jr.)** |

# BIOGRAPHY

Mohamed Sayeed was born in Bangalore, India. He completed his Bachelor of engineering degree in civil engineering from Bangalore University in 1990. He worked as a civil engineer in different capacities, as a consultant and entrepreneur. He completed his Master of engineering degree in structural engineering in 1997 from Bangalore University. He worked as a structural engineer for a consulting firm before leaving for higher studies. He joined the Ph.D. program in computer aided engineering in civil engineering at North Carolina State University, Raleigh in the fall of 1999. His research interests are in parallel computing, parallel performance analysis, numerical analysis, optimization and finite elements.

# ACKNOWLEDGEMENTS

I would like to thank all my committee members, teachers, colleagues and friends for all their help and understanding.

I would like to thank my advisor and co-chair Dr. G. Mahinthakumar for his valuable advice and support. He is the force behind this research and was always available for discussion.

I would like to thank Dr. John W. Baugh for being my co-chair and being supportive. I thank Dr. Ranji S. Ranjithan for being on my committee and for the valuable discussions we had on the optimization approaches. I appreciate Dr. Abinav Gupta for being my committee member.

I thank Emily Zechman for her help on MGA approach and Baha Mirghani for creating an animation of the GA performance.

I thank Dongju Choi and Leesa Breiger from SDSC, for doing the multi-cluster runs on the TeraGrid.

I wish to acknowledge North Carolina Supercomputing Center, Oak Ridge National Laboratory, National Center for Supercomputing Applications, San Diego Supercomputer Center and National Energy Research Supercomputing Center for providing the supercomputer resources necessary for this work.

I acknowledge the support indirectly provided by the PERC (Performance Evaluation Research Center) project (Dept. of Energy's Scientific Discovery through Advanced Computing Program) subcontracted through ORNL from September 2001.

This work was partly supported (Oct 2003- present) by National Science Foundation (NSF) under Grant No. **BES- 0238623**.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1 – INTRODUCTION

Inverse problems that are governed by large-scale partial differential equations (PDE) require significant computational resources and can be several orders of magnitude more computationally challenging than the corresponding forward problem (i.e., prediction) because repeated solutions of the forward problem are necessary. These problems are particularly challenging in subsurface contaminant characterization because the forward model can consist of several coupled large-scale nonlinear partial differential equation systems that can vary in three-dimensional space and time. Recent advances in search techniques such as genetic algorithms (GAs) and emergence of advanced computing resources such as the computational grid (networked supercomputers), have opened up new possibilities for solving these inverse problems. The most common approach for solving subsurface characterization inverse problems is the use of gradient-based optimization methods. While these methods are very powerful and are appropriate in many situations, they lack the generality of non-gradient approaches (e.g. genetic algorithms, simulated annealing) and are less suited for emerging parallel computing environments.

GAs are popular global search procedures for discrete as well as continuous problem domains, but yet under-explored for solving these problems. The GA search process is enhanced by the use of local search (hybrid approach). The main themes for the proposed research are to investigate: (i) hybrid optimization approaches (global and local searches) and, (ii) parallel computing techniques to solve inverse problems (in the groundwater area).

The thesis has seven chapters. This chapter gives an overview of inverse problems with examples, solution techniques for inverse problems and the application of high performance computing. The 2nd chapter discusses previous research on optimization-based approaches for subsurface characterization problems. Hybrid optimization approaches using GAs and different local search techniques are discussed in the 3rd chapter. The discussion on parallel implementation is covered in the 4th chapter. The 5th

chapter presents the results of testing and validating the hybrid optimization methodology. Information on parallel architecture, parallel performance results and grid-computing results are discussed in 6th chapter. The 7th and final chapter summarizes research contributions and topics for further research work. In Appendix – A, the noisy-GA approach is investigated for problems with parameter uncertainty. In Appendix B, a preliminary investigation of the modeling to generate alternatives (MGA) approach is carried out for addressing the non-uniqueness problem.

### 1.1 Inverse Problems

An inverse problem involves the estimation of certain quantities based on indirect measurements of other dependent quantities. This research focuses on groundwater inverse problems. Inverse problems also arise in other diverse fields such as geophysical exploration, medical imaging, non-destructive evaluation, inverse heat conduction or diffusion problems, and signal processing. In signal and image processing one tries to recover the original (uncorrupted) signal from the filtered signal with noise. Use of computer aided tomography and magnetic resonance imaging in medical diagnosis, has lead to the development of algorithms for the inversion of the Radon transform. The exploration of oil is often facilitated by knowledge of the electrical conductivity structure of a rock formation. The conductivity itself is ascertained from establishing a magnetic field in the rock formation by measuring the induced currents. Seismic exploration yields measurements of vibrations recorded on the surface. These measurements are only indirectly related to the subsurface geological formations that are to be determined.

In parameter identification, spatial and/or temporal parameters appearing in, e.g., partial differential equations, are determined from measurements of the solution, either in the whole domain, or on the boundary only. Inverse heat conduction or diffusion problems for determining the boundary heat flux, and inverse scattering problems to determine the shape or the location of the obstacle based on measurement of waves [Santamarina and Fratta 1998]. Many problems in solid mechanics like identification of cracks singularities, identification of material constants, separation of different energies in solids, determination of residual stresses etc. may be considered inverse problems.

Other inversion examples are: loading a concrete specimen and measuring its deformation to determine material properties such as Young's modulus and Poisson ratio, or the deflection of the bridge is measured to access the condition of cables and deck [Bui 1993]. A performance based dynamic structural design approach using inverse problem formulation has been developed [Takewaki 2000].

A number of groundwater problems such as estimating hydraulic conductivity distributions, biological activity zones (BAZ), dense non-aqueous phase liquids (DNAPL), or contaminant sources location and release history have been solved using the inverse problem formulation. A very good reference for inverse problems in ground water modeling is by Sun, 1994.

While common enough in practice, groundwater problems such as these are notoriously difficult to solve because of several factors namely, insufficient observation data, error in model or input data, etc. During the past several year's significant developments in probability and control theory methods have helped solve such complex inverse problems [Sun 1994]. Most of the inverse problems are characterized by an unusually high sensitivity to perturbations (deterministic as well as stochastic) in the data so that a small change in the measurements results in disproportionate error in the recovered signal. Techniques such as regularization methods have been developed to deal with this Ill-Posedness. Thus, solving such inverse problems is not only numerically challenging, but they also demand large computational resources.

### 1.2 Computational methods for solving inverse problems

Any mathematical model or system should be able to solve a forward problem in order to solve an inverse problem. However obtaining a forward solution accurately does not guarantee that the inverse solution is also going to be accurate. This is the case because inverse problems are generally ill posed, which means the solution may be non-unique, non-existent and unstable. The primary reasons are observation error, model structure error and the insufficient quantity and/or quality of observation data. If these difficulties are not considered, a satisfactory inverse solution can never be obtained by purely changing the performance criteria and/or optimization algorithm.

As stated earlier, before solving an inverse problem a mathematical model for solving the forward problem should exist. So a concise discussion on mathematical modeling of forward problems is provided next. Mathematical models can be broadly classified in to four types: (i) Deterministic models and stochastic models depending on whether random variables appear in the model; (ii) Linear models and nonlinear models depending on whether the equations are linear or nonlinear; (iii) Stationary and dynamic models, depending on if the time variable is included and, (iv) Lumped parameter models and distributed parameter models depending on whether the space variables are included. In groundwater modeling distributed parameter modeling is preferred because it is more general, more accurate and more suitable for the planning and management of groundwater resources. A distributed parameter model is often described by a PDE or a set of PDEs and may be classified in to one or several of the above types. Generally, the distributed parameter model involves the following components: it may have both spatial and temporal properties, system parameters that characterize the geometry and/or physical nature of the system, initial condition of the system described by one or more subsidiary conditions, control variables representing excitation of the system such as pumping, artificial recharge etc., and state variables that describe the state of the system, such as head, concentration etc. Solving a forward problem implies to determine state parameters when the time-space region, system parameters, subsidiary conditions and control variables are known. The solution can be obtained analytically or using a numerical approach. Analytical solutions are however available only for simple problems. The solutions can be obtained by superposition of fundamental solutions, separation of variables, Laplace transformation, Fourier transformation and other integral transformations. Numerical methods use discretization of the time-space domain in to elements and nodes. The governing system of equations (PDEs) is then discretized and replaced at these nodes by a system of algebraic equations. The solution of the equations can be obtained using different methods such as finite difference methods, finite element methods, boundary element methods and their variants and hybrids, including the hybrids of numerical and analytical solutions [Sun 1994].

If a mathematical model is available to solve a forward problem then it can be used in an iterative fashion to solve the inverse problem. However the accuracy of the forward model is no guarantee for the quality of the solution obtained in the inverse approach. This is normally referred to as the Ill-Posedness of the inverse problem. Three main criteria describe the Ill-Posedness of the inverse problem

*Non-existence*: The solution may not exist for the observation data.

*Non-uniqueness*: The solution may not be unique because different conditions may give same observation data.

*Instability*: A small change in input results in a disproportionate output.

The above three properties are associated with many of the inverse problems encountered in real applications.

As stated earlier inverse problems can be solved in both deterministic and stochastic frameworks by direct or indirect methods. Indirect methods require repeated solutions of the forward problem. Many of the methods solve the problem by using an output least squares criterion, which is a measure of the error between the measured and the computed values. In general, direct methods can only be used for solving inverse problems governed by linear system of equations. The commonly used direct methods are the matrix method and the linear programming method. The matrix method reformulates the linear least squares problem as a set of matrix equations and solves it directly. The matrix method normally yields highly ill conditioned matrices and is very sensitive to measurement errors. Linear programming methods can be used to solve inverse problems governed by linear equations. Direct methods have very limited applicability to groundwater inverse problem due to the distributed nature of the parameter space, non-linearity in the governing equations and measurement errors.

The work here focuses on indirect methods and these are generally optimization-based approaches. They can be grouped into two broad categories: gradient and non-gradient based approaches. The commonly used gradient methods are: steepest descent,

Levenberg-Marquadt, Gauss-Newton, conjugate gradients and sequential quadratic programming methods. The gradient-based approaches use the gradients to find the solution and hence require the decision space to be continuous (smooth). Thus, the gradient methods are not suitable for discrete optimization problems. However, the gradient methods tend to converge faster than non-gradient based approaches if: (i) the objective function is continuous and differentiable, (ii) the search space is fairly smooth and not convoluted and (iii) the quadratic assumption is valid near the optimum (for some approaches).

Non-gradient methods start with one or more initial guesses and follow a set of rules to move towards the solution. These optimization methods can be global or local. Examples of global approaches are genetic algorithms (GAs) [Holland 1975, Goldberg 1989], simulated annealing, particle swarm optimization (PSO) [Kennedy and Ehart 1995], and the DIRECT method [Floudas and Pardalos 2001]. Examples of local non-gradient approaches are, Nelder-Meade simplex method, Hooke-Jeeves pattern search method, and Powell's method of conjugate directions. Non-gradient based techniques such as GAs offer great flexibility in problem formulation and can handle discontinuities in the search space. Furthermore, they can generally explore a larger search space than gradient-based approaches. As stated earlier GAs are generally preferred for discrete or discontinuous domains and is the global method of choice for this research. GAs can also be used effectively (real GAs) for problems that are not inherently discrete. The focus of this research is to establish a parallel optimization framework for solving groundwater inverse problems.

### 1.3 Role of high performance computing

In recent years high performance computing (HPC) has opened up new opportunities for modeling and understanding complex systems (e.g., groundwater modeling). It is driving the frontiers of science to new levels and the need for additional computing power for solving complex and larger problems keeps growing. Computation is becoming an equal partner with theory and experimentation in the advancement of science. It has opened new avenues for understanding complex (natural) phenomenon,

and offered challenges for the development of new and efficient mathematical models. High performance computing (HPC) offers the ability to compute solutions to problems not possible on desktop machines. It provides the ability to: (i) use larger and more detailed computational grids, (ii) develop more complete computational models of physical processes, and (iii) perform large number of simulations under different conditions. HPC is enabled by large parallel computers, powerful vector processors, or a network of individual workstations. These parallel computers link processors or individual workstations together to increase computational power and use high-speed, large-memory-capacity computers. Algorithmic improvements with state-of-the-art numerical techniques such as adaptive meshing, multigrid, and particle methods are needed in the next generation of high-performance simulation codes (e.g., groundwater).

For problems that require significant computational resources, parallel computing on supercomputers or newly emerging "*grid computing*" on a network of geographically distributed heterogeneous supercomputers connected by high-speed network can be used. Also, distributed computing on cluster of networked workstations can be helpful. Parallel computing requires knowledge of parallel machine architectures for efficient implementation of parallel applications and efficient parallel numerical algorithms. One or more parallel programming tools such as MPI [Message Passing Interface, Gropp et al. 1999; Dongarra et al. 1994], OPENMP [www.openmp.org], PVM [Parallel Virtual Machine; Giest et al 1994], etc., are available on high performance computers for use. Currently MPI has emerged as the de-facto standard and is available on almost all supercomputers. Parallel codes using MPI are portable. MPI implementations supporting many languages including popular HPC languages such as Fortran and C are available.

Writing efficient parallel programs to take full advantage of a high-performance multiprocessor requires a new way of thinking (programming methodology) and effective software tools to deal with the inherent complexities. While automatic parallelization tools such as Forge or OpenMP (for shared memory) are now commercially available. They still require substantial programmer intervention either in adding compiler directives or restructuring code. Without this intervention, the resulting compiled code usually performs poorly.

Parallel programming is notoriously difficult because it lacks the single thread of control that one would have in a conventional serial program. In addition, the data used in parallel computation are probably spread across a number of distributed computer systems. This is done to capitalize on locality by leveraging faster local data accesses against more costly remote data accesses. Often, the data “decompositions” that make a parallel program the fastest are the ones that are the most complicated [Kohl 1997].

This research requires high performance computational resources as it solves the forward problem repeatedly. The forward problem (transport) simulator [Mahinthakumar and Saied 2001] as will be known in a later chapter, is a highly computationally intensive FEM code, solving large-scale partial differential equations. Hence, a parallel optimization framework is developed to handle complex 3D groundwater inverse problems.

# CHAPTER 2 – RELATED RESEARCH IN GROUNDWATER INVERSE MODELING

Much of the previous work in groundwater inverse modeling has been in model calibration aimed at fitting a few parameters (determine a few parameters in the forward model based on observation data). With the development of sophisticated forward models in recent years inverse modeling can now be used for obtaining detailed information about the subsurface. This subsurface information is critical to the efficacy and cost efficient groundwater management strategies. The following sections provide an overview of previous research using optimization approaches for groundwater modeling.

## 2.1 Non-heuristic approaches

A large amount of research has been done in using gradient-based approaches for solving groundwater inverse problems (e.g. Gorelick et al 1983, Wagner 1992, Sciortino et al. 2000, Mahar and Datta 2000). Gorelick et al (1983) used least squares regression and linear programming for solving a hypothetical two-source groundwater contaminant problem. They assumed a linear model and incorporated the solute transport model as constraints in a response matrix approach [Gorelick 1982; Gorelick and Remson 1982b] for solving. Two hypothetical scenarios were studied: (1) locating unknown pollutant sources under steady state from concentration data collected at a few well locations, and (2) reconstructing the release history and location of a source using a complex two dimensional (2-D) transient system with several monitoring wells. For the steady state case there were more unknowns than the number of constraining equations. A mixed integer programming method with additional restrictions was used. The results obtained were spurious and detracted from true values. For the transient case both methods identified the pollution source and the disposal episodes, but contained some errors in determining the disposal flux magnitudes. The method is restricted to cases where data are available in the form of break through curves.

Given the importance, groundwater source identification and release history reconstruction problems have received a lot of attention. A variety of methods like the

minimum relative entropy approach - MRE [Woodbury and Ulrych 1996, Skaggs and Kabala 1998, Woodbury et al. 1998], Tikhonov regularization - TR [Skaggs and Kabala 1994, Liu and Ball 1999], constrained nonlinear optimization [Mahar and Datta 2000, 2001], and Levenberg-Marquadt minimization [Sciortino et al. 2000] have been used. A paper by Atmadja and Bagzoglou (2001), gives a state of the art report on mathematical methods for groundwater pollution source identification.

The regularization methods such as TR and MRE try to alleviate the problem of ill-posedness and computational complexity of inverse problems. For example, the TR method removes discontinuity in the solution space by smoothing the objective function (either directly or indirectly). This addresses the problem of instability (small changes in decision variables lead to large variations in objective function) and non-uniqueness (multiple solutions) by forcing the convergence to the 'simplest' solution (solution that has the smoothest structure). There is no guarantee, however, that this is the best solution. MRE method treats each element of the release history as a random variable. The MRE inversion is a method of statistical inference. It constructs a probability density function (pdf) for the random variables representing the solution based on prior information and measurement data; the solution is the mean of this pdf. Neupaur et al. (2000) made a comparative study of TR and MRE methods for different source release history recovery problems. The results show the quality of solutions obtained by these methods to be input and problem dependent.

Sciortino et al. (2000) solved an inverse problem to identify the location of dense non-aqueous phase liquid (DNAPL) pool in a saturated porous medium under steady flow conditions. Levenberg-Marquardt method is used to solve the least squares minimization problem for the identification of the location and the geometry of the DNAPL pool. They used the method to minimize three types of residuals: ordinary residuals, weighted residuals with weights equal to the square of the inverse of the observation, and weighted residuals with weights obtained by adding a constant term to the observed concentrations. They observed the results being sensitive to the location of observation wells and the type of residual minimized. The procedure was not robust, since it did not guarantee the convergence to global minimum, as the inverse problem solved is non-unique and non-

convex even in the absence of observation errors. Also, the results produced by the model varied according to the number and location of the observations.

Datta et al. (1989) developed an expert system using statistical pattern recognition techniques to identify sources of groundwater pollution for hypothetical example problems. The model developed by Gorelick et al. (1983) was used as a preliminary screening model within the expert system. Performance of their method was found encouraging in general for the example problems and specifically good under conditions of missing observed-concentration data. Bagtzoglou (1992) presented the application of a random walk based model for identification of pollutant sources in groundwater. Wagner (1992) combined nonlinear maximum likelihood estimation with ground-water flow and solute transport simulation to simultaneously estimate the aquifer parameters and a distributed pollutant source term.

Mahar and Datta (1996) combined the concepts of optimal identification of pollutant sources with the optimal design of a ground-water-quality monitoring network for the source identification problem. They used a nonlinear optimization model with embedded flow and transport simulation constraints for optimal source identification. Crank-Nicolson and implicit finite difference forms along with appropriate initial and boundary conditions, are embedded as constraints in this identification model. The solute transport equation is discretized using central difference in both space and time, and the steady state flow equation is discretized using central difference in space. An integer-programming based model was used for the monitoring network. The methodology follows three steps with the first step utilized for preliminary identification of pollutant sources (magnitude, location and duration of activity) based on observed concentration data from arbitrarily located existing wells. The preliminary identification results from first step (source locations) are used to obtain an optimal monitoring network in the subsequent time periods. In the third step the designed monitoring well locations are utilized for more accurate identification of pollutant sources. The performance of their proposed methodology was applied to a hypothetical 2-Dimensional homogeneous, isotropic, and saturated aquifer for identifying three potential source locations. They concluded the methodology using optimal network design enhanced the applicability for

real-life scenarios when initially the potential source locations are unknown and where measurements and parameter estimates are erroneous and/or uncertain. However their study did not address the effect of parameter uncertainty and only a limited numbers of cases were tested.

Following their work in 1996, Mahar and Datta (1997, 2000 and 2001) solved different types of source identification problems using non-linear programming approach. They used finite differences and the finite difference equations as constraints to model two-dimensional forward flow with steady state or transient and transport problems. The governing equations for the physical processes were embedded in the optimization model. They solved the resulting nonlinear programming problem by a quasi-Newton constrained optimization method. The model is tested for 2 or 3 potential source locations, both with and without perturbed observation data and also with data gaps. The model was not very robust because of the nonconvex and nonunique nature of the inverse problem and because the results were dependent on the initial guess provided to the method. They extended the model to simultaneously estimate the aquifer parameters (2001).

**2.2 Heuristic approaches**

While heuristic techniques such as GAs have been used widely for groundwater management problems [McKinny and Lin 1994; Wang and Zheng 1998], or for optimal placement of monitoring wells [Cieniawski et al. 1995, Ritzel et al. 1994, Huang and Mayer 1997, Katsifarakis et al. 1999] they have not been used as extensively for solving groundwater inverse problems.

Aral and Guan (1996) used GAs for searching groundwater pollution sources. They proposed an improved GA (IGA) procedure in which members of the population are generated in a restricted solution space and equality constraints are satisfied for a subset of decision variables. They applied the method to contaminant source location, leakage rate and release period identification problems for a hypothetical confined aquifer system by posing it as an optimization problem with equality constraints. They reported results for a continuous leakage problem and a time dependent leakage problem

with data gaps, for both without noise and random noise in the synthetically generated reference data cases. Based on their study they concluded that their IGA provides an efficient and robust means for solving any quadratic optimization problems with linear equality constraints.

Katsifarakis et al. (1999) used boundary element method (BEM) approach for modeling groundwater flow and transport and coupled it with GA to solve common groundwater management and inverse problems. The application examples studied are (1) determination of transmissivities in zoned aquifers, both with and without field measurement errors, (2) minimization of pumping cost from a group of wells, and (3) hydrodynamic control of a contaminant plume. They claimed that the method performed satisfactorily.

Mahinthakumar et al. (1999) used GAs for identifying zones of biological activity in the subsurface. They used a parallel computing environment for solving, as repeated three-dimensional finite-element forward function evaluations are required for every individual in a GA population. The simulations performed showed the effective application of GAs in inverse groundwater modeling.

Smalley et al. (2000) presented a risk based in situ bioremediation design using a noisy genetic algorithm. The model couples a noisy GA with a numerical fate and transport model and an exposure and risk assessment model that translates the predicted concentrations into estimates of human health risk, allowing risk-based criteria to be considered in the corrective action design. In the risk based corrective action (RBCA) design technique the uncertainty and variability in the parameters associated with groundwater simulation and exposure modeling significantly influences the risk assessment. The noisy GA is used for solving the management model with parameter uncertainty and variability for finding robust solutions. In the model, uncertainty is incorporated by the spatial distribution of hydraulic conductivity and the variability of exposure parameters. The noisy GA uses sampling to reduce noise from fitness evaluations in noisy environments. Unlike Monte Carlo simulation modeling, however, which requires that numerous samples be drawn from probability distributions to obtain

reasonably accurate results, noisy genetic algorithms perform best without extensive sampling [Miller and Goldberg, 1996]. On the basis of Aizawa and Wah's (1994) sampling strategy for noisy GA's, the following approach was used. For the first four generations sampling size was set to 5 and was increased by five sample sets every four generations. After twelve generations fittest four designs from the previous four generations (i.e., 9-12) were tested by simulating each with five hundred sample sets. If any of the four designs are successful in meeting the risk criteria for at least 90% of the realizations, then sample size was not increased and the optimization process continued for four more generations before termination. Otherwise, the sample size is increased by five sample sets in the same manner described above with a test for four fittest designs from the previous four generations every four generations until successful termination or until a maximum number of generations was reached. Based on a case study, the authors concluded that noisy GA was capable of generating highly reliable designs from relatively small number of sample sets and efficient for computationally intensive groundwater management models. The authors suggested the need for further investigation of sampling strategies and termination criteria that affect GA efficiency, the values of the decision variables for the optimal design, and the reliability of the optimal design.

Yoon and Shoemaker (2001) proposed an improved real-coded GA (RGA) for bioremediation. They proposed a new technique for crossover and selection (replacement) process of GA. The RGA developed was tested for two hypothetical aquifers developed by Minsker (1995). Based on the study they reported RGA was efficient and computationally accurate than binary encoded GAs used in previous groundwater research.

Aral et al. (2001) proposed another combinatorial approach called progressive genetic algorithm (PGA) for nonlinear optimization. The method uses two steps: (i) iteration and, (ii) search. In the iteration step ground-water simulation models are run to generate an approximate optimization model for the subdomain. The solution domain is divided in to subdomains depending on the number of sources searched. It transforms the implicit nonlinear optimization problem in to a series of approximate optimization

problems with explicit linearized constraints, which are then solved by GA. In the search stage GA is used to search for local optimal solution with in the subdomain and closer to the previous solution. The authors applied the methodology to test several groundwater applications for a single source in a heterogeneous unconfined aquifer system. They solved the release history reconstruction problem for known source location, unknown source location, and release history with some observation data missing (gap). Based on their study of the above problems they claimed PGA technique as robust and computationally efficient. Several observations about the factors that influence the solution of the source identification problem were also reported.

Giacobbo et al. (2001) solved the inverse problem of parameter estimation by GA for groundwater contaminant transport. They analyzed the sensitivity of the model to the unknown input parameters from the speed of convergence and stabilization of the GA. They showed that the GA evolves towards convergence by stabilizing first the most important parameters (parameters that influence the model output most) and later the parameters that influence the output less.

Hilton and Culver (2003) used genetic algorithm for groundwater remediation design under uncertainty. The authors proposed a new modified GA called robust GA and compared the performance with basic GA and noisy GA. The way parameter uncertainty is introduced in the model differs from noisy GA. It includes uncertainty by utilizing a measure of on-going performance to evaluate the robustness and reliability of a possible design. Thus, robust GA incorporates uncertainty into the optimization procedure, rather than in the objective function evaluation, as done in the noisy GA. The robust GA is a multiple realization technique; however, it uses a single but different realization in each generation. The overall fitness of the string is based on the performance over multiple generations. The fitness of the string is a function of cost (objective function value) and “age”. The age indicates the number of generations the string survived. A new rank fitness based on these parameters is computed and is used during selection. Two test cases for groundwater remediation design using homogeneous and heterogeneous hydraulic conductivities are reported. The authors showed that the robust GA performed as well as the noisy GA, but by using fewer number of objective function evaluations.

Shieh and Peralta (2003) used parallel recombinative simulated annealing (PRSA) for optimizing in-situ bioremediation system design. PRSA is a global optimization algorithm with convergence properties of simulated annealing (SA) and parallelism of GA. The proposed model uses BIOPLUME II model to simulate aquifer hydraulics and bioremediation, and PRSA to search for an optimal design. They solved for an optimal pumping (extraction/injection) strategy that minimizes total system cost, reduces contaminant concentration to the cleanup standard, and prevents contaminant plume migration. For the test problem the PRSA approach performed better than SA and GA. They claimed the approach to be efficient and flexible for optimizing system installation design and time-varying pumping.

### 2.3 Hybrid optimization approaches

Very limited amount of work has been done to date using hybrid optimization approaches for solving groundwater inverse problems [Heidari and Ranjithan 1998, Pan and Wu 1998]. Heidari and Ranjithan used a hybrid GA- truncated Newton search for a two-search approach for a two dimensional hydraulic conductivity estimation problem. Pan and Wu (1998) used a simulated annealing – simplex approach for estimating unsaturated flow parameters for a one-dimensional column experiment.

More recently, Espinoza et al. (2003) developed self-adaptive hybrid GA (SAHGA) and non-adaptive hybrid GA (NAHGA) and compared their performance with simple GA (SGA) for a groundwater remediation problem. The authors also presented a methodology for selecting the HGA parameters and population size for optimal performance. They coupled SGA with a local search approach (random walk algorithm). They discussed several performance and modeling issues relating to hybrid GA such as local search frequency, probability of local search (LS), and number of local search iterations. The selection of correct NAHGA parameters is crucial for its performance, and the “necessary trial-and-error process for parameter evaluation makes the application of NAHGA impractical”. Even though the proposed NAHGA and SAHGA methods achieved convergence for the test case with 75% and 85% fewer function evaluations than SGA, obtaining right parameter settings was still challenging with a trial and error

process. In general the hybrid approach performed better. Also, SAHGA is more robust than NAHGA because local search is applied only when it is necessary and the performance does not change for a broad range of different parameter values.

The present research work seeks to enhance existing knowledge. It investigates the applicability of hybrid optimization methodology for solving complex inverse problems in a parallel computing environment. Even though the problems tested in this research is in subsurface characterization the generality of the optimization approaches will facilitate easy extension to other areas. The rationale for using hybrid GA-LS approach is that GA being a global search technique will take the solution closer to the global optimum and the local searches can do the fine local tuning. Other related topics of interest such as noisy GAs and Modeling to Generate Alternatives (MGA) are also investigated.

# CHAPTER 3 – OPTIMIZATION METHODOLOGIES

This chapter describes the optimization methodologies used in this research. The research uses hybrid optimization approaches for solving inverse problems. The sections that follow discuss GA methodology (both binary/integer and real GA) and the four local search methods, Nelder-Meade simplex (NMS) method, Hookes and Jeeves pattern search (HKJ) method, Powells conjugate directions (PWL) method, and Fletcher and Reeves conjugate gradient (CG) method. Figure 3.1 shows the optimization algorithms implemented in this research.

## 3.1 Overview of Genetic Algorithms

A number of variants of GAs have been developed, researched and extensively used for a wide variety of applications. GA's optimize using a search process that emulates natural evolution. In a GA, a potential solution to a problem (i.e., a possible set of values to represent the unknown parameters) is represented as a vector (an 'individual'). This vector traditionally consists of binary values, although real numbers are being used increasingly. Potential solutions and parameter values are analogous to organisms and genes, respectively. The GA starts the process of searching for good solutions with a set of these potential solutions, called a population, which is often generated at random. The performance of each solution is characterized by a fitness value. In the context of inverse problems, fitness is inversely proportional to the difference, or residual, between computed and observed values. Calculating fitness for each solution requires a forward solve.

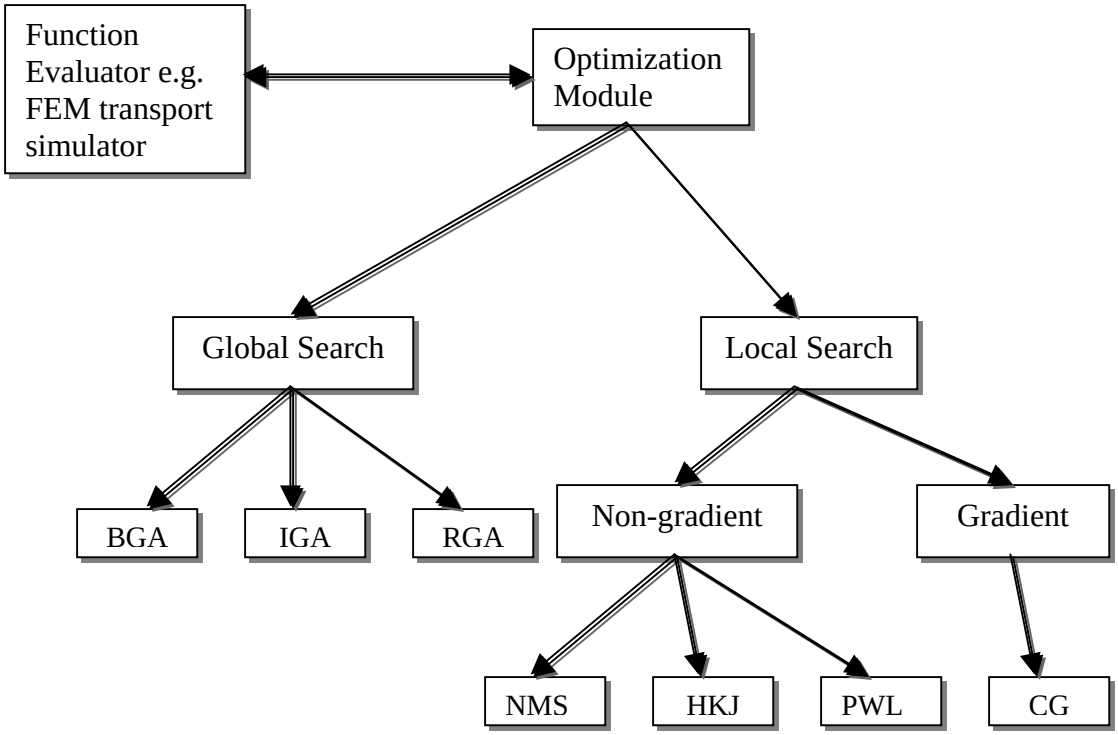



**Figure 3.1 Different optimization algorithms available in the module for solving inverse problems.**

During the GA search process, the population is subjected to several probabilistic operators that are analogous to natural selection, mating (including genetic recombination), and mutation. In the selection step, pairs of solutions are selected for reproduction from the population, with fitter individuals being selected more frequently. Each pair of solutions may then undergo mating, or crossover, in which their vectors are recombined to form new solutions, which are placed into a new population. The selection and mating steps continue until the new population is the same size as the previous population, which is then discarded. After the new population has been generated, mutation is used to modify a small number of genes in the population. This step introduces new traits that may not have been present in the initial population. Mutation may also reintroduce good traits that may have been lost through the probabilistic selection operator. An additional operator called elitism is used in most GA applications. Elitism, which generally occurs after mutation, is used to guarantee that the best individual in a population is not lost.

The flowchart for GA is shown in figure 3.2. The steps of evaluating the population, selection, mating, mutation, and elitism constitute one iteration, or generation. Because fitter solutions are more likely to be selected for mating, the incidence of good traits in the new population generally increases with each additional generation. Crossover serves to test these traits in many different combinations. GA schema theorem predicts the frequency of good traits (and good combinations of traits) to increase exponentially as new generations are formed. As this occurs, the GA converges to increasingly better solutions. Improvements in fitness, however, diminish as the population diversity decreases and the population converges towards a good solution. Stopping criteria such as "10 generations without improvement" and a minimum population diversity are often used to terminate the algorithm when improvements are sufficiently small and infrequent. These concepts are well described in many texts, including Goldberg (1989), Davis (1991), and Michalewicz (1996).

The use of GAs for optimization problems in groundwater management and remediation has been abundant (see section 2.2). Two main factors that have limited the use of GAs for these and other problems are: (i) GAs have been characterized as being highly computationally intensive, and (ii) GA performance is a function of the search parameter values (e.g., population size, mutation rates etc.), the best values of which cannot be determined a priori. While these are legitimate concerns, each can be addressed to some extent. For example, the use of parallel computing can address concern (i). Also, the argument that gradient-based approaches are more computationally efficient is not always true. For problems with a large number of decision variables, computational requirements associated with the calculation of gradients can far exceed that of the GA search process. Limitation (ii) is being addressed through recent research to identify robust operator implementations and parameter settings [Goldberg and Shastry 2001, Lobo and Goldberg 2001, Reed et al. 2000b]. The guidelines developed through this research are resulting in more robust GA formulations, effectively reducing or eliminating trial-and-error experimentation with alternative GA parameter settings.

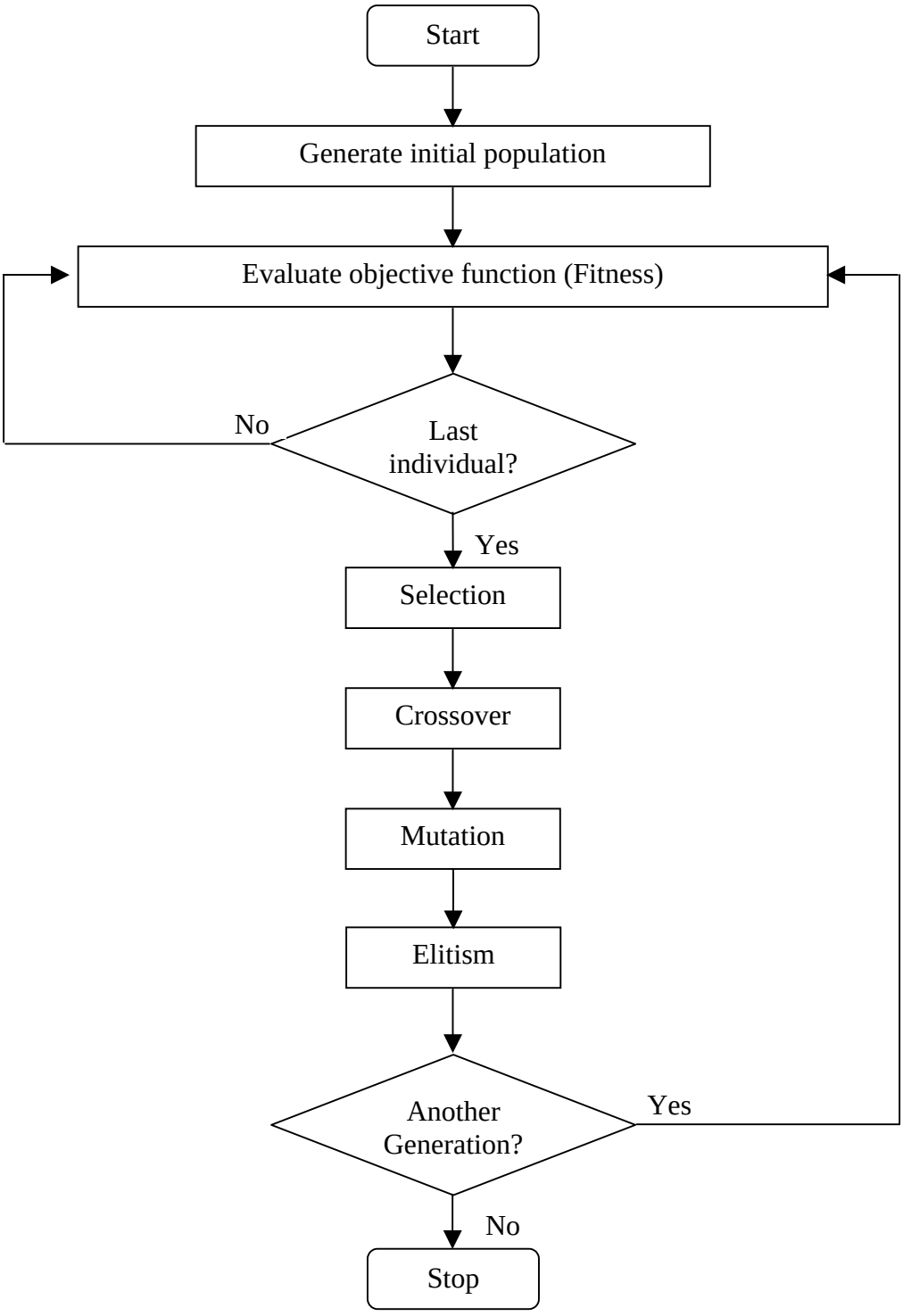



**Figure 3.2 Flowchart of genetic algorithms (GA).**

### 3.1.1 *Genetic Algorithms for inverse modeling*

As described earlier, GA starts with a set of potential solutions, or population and the performance of each solution is characterized by a fitness value. In the context of inverse problems, fitness is calculated using a forward solve and is inversely proportional to the difference between computed and observed values. One of the drawbacks of GA is that it is computationally intensive if the fitness evaluation (or objective function) is expensive. In this application, the objective function to be minimized is the root square error (RSE) between the observed and the computed output concentration signals at a few selected points in the domain. RSE is given by

$$RSE = \sqrt{\sum_{i=1}^{n} (C_i^{obs} - C_i^{calc})^2}$$

Where $C_i^{obs}$ is the observed concentration and $C_i^{calc}$ is the calculated concentration at the observation points '*i*' the observation number and '*n*' the total number of observations. In order to compute the output signals for each individual in a GA operation, a forward transport simulation is performed. Because the time intensive fitness calculation for each individual in a generation can proceed independently, GA's are amenable for use in a parallel or distributed computing environment. Whereas the use of GAs has long been popular for groundwater optimization problems, only recently GAs are being used for groundwater inverse modeling [Mahinthakumar et al. 1999, Aral et al. 2001, Giacobbo et al. 2002]. Three types of GAs namely binary, integer and real have been implemented to study the different classes of problems tested in this research. Some of the problems such as biological activity zone (BAZ) are inherently discrete. Binary/Integer GA is used to solve biological activity zone (BAZ) identification problem. The source identification problem is solved by BGA, RGA, and hybrid GA-LS approaches. The source release history problem is solved by RGA and hybrid RGA-LS approaches. More details on the problem and results are given in the next chapter. The subsections that follow give an overview of the implementation of these methods.

### 3.1.2 *Binary/Integer GA Implementation*

The BGA and IGA implementations are required for the BAZ identification problem (see section 5.2). A slightly enhanced version of the Simple Genetic Algorithm (SGA) presented in Goldberg (1989) is used for the BGA and IGA implementations. The primary modifications are elitism (always retain the best solution in the new population), additional selection procedures (tournament selection in addition to the original roulette wheel selection), additional crossover strategies (uniform and multi-point crossovers in addition to the original simple crossover), and support for both binary and integer encoding. An adaptive mutation operator is implemented so that the mutation probability can be progressively reduced when the RSE of the best individual drops below an arbitrary threshold value (if the correct solution is known a priori then this threshold value can be set close to zero). A *restart* option is implemented so that the simulation can be restarted from the last completed generation.

In the simulations performed the following steps are involved:

(1) Encode the unknown zone locations as binary or integer strings.

(2) Generate an initial random ensemble of strings (with a user-defined bias) equal to the number of individuals in a population or population size.

(3) Perform transport simulation for each individual by decoding the strings into zone locations.

(4) Compute RSE for each individual by computing the difference between the observed (stored in a file) and computed output signals.

(5) Select the individuals that perform best (those giving a smaller RSE) using an appropriate selection strategy and mate the strings randomly (using an appropriate crossover strategy – single point, uniform, multiple point) to produce the next generation for individuals.

(6) Repeat steps 3 – 5 until convergence or up to prescribed maximal number of generations.

(7) If convergence is not achieved within the prescribed number of generations, then either the zone locations for the best-performing individual of all the

generations or the probability distribution of the entire population at the end of simulation can be chosen as the optimal solution.

### 3.1.3 *Real Genetic Algorithm (RGA) Implementation*

Real GA follows the same steps as BGA/IGA. As stated before, for RGA decision variables are represented as a vector of real variables (within some bounds). For source identification and release history reconstruction problems, RGA is more suitable as the decision variables (source location coordinates, concentrations, and time history of concentration release) are inherently real. The real encoding can represent large domains with a smaller string length when compared to its binary representation without sacrificing the precision of numbers. Also increasing the number of bits considerably slows down the algorithm [Michalewicz (1996)]. The concepts and operators (selection, crossover, mutation and replacement) are very similar to BGA.

The RGA implementation has four different crossover strategies simple, uniform, arithmetic and heuristic crossovers. In arithmetic crossover a linear combination is performed using the following expression, if $x_1$ and $x_2$ are crossed then the two offsprings will be $x_1' = rx_1 + (1-r)x_2$ and $x_2' = rx_2 + (1-r)x_1$, where '*r*' is a random number generated between 0 and 1. In heuristic crossover a single offspring is produced from two parents $x_1$ and $x_2$ according to this rule: $x_3 = r(x_2-x_1) + x_2$, where '*r*' is another random number generated between 0 and 1, and parent $x_2$ is not worse than $x_1$. The algorithm uses a combination of crossover strategies instead of just one depending on the random number generated. The following strategy is adopted, if $r \leq 0.1$ – simple, $0.1 > r \leq 0.3$ – uniform, $0.3 > r \leq 0.8$ – arithmetic, and $0.8 > r \leq 1.0$ – heuristic crossover. It should be noted that these values were obtained after initial experimentation and should not be construed as the best. These values consistently performed well for our problems. A combination of crossovers is chosen as it inherits the best properties of individual crossovers. Also either uniform or non-uniform mutation operators can be used. In non-uniform mutation one of the elements $x_k$ of the vector $x$ (parent) is selected, the offspring $x' = \{x_1, ...., x_k', ...., x_q\}$. The element $x_k'$ is calculated as follows

$x_k^{'} = x_k - dt$ and $dt = (x_k^u - x_k)(r(1-a))^b$,

Where, $x_k^u$ - upper bound for the element of the chromosome string, *r*- random number between 0 and 1, *a* is the ratio of generation number to maximum number of generations and *b* is a system parameter determining the degree of non-uniformity (assumed to be 1). For all the simulation experiments performed using real GA, non-uniform mutation is used. Additional information on real GAs is available in many books on GAs such as by Michalewicz (1996).

### 3.2 Local search methods

Non-gradient based unconstrained optimization methods are implemented for local searches. These methods were selected because they do not require the computation of gradients and are easier to program. The methods are: (1) Nelder-Meade Simplex, (2) Hooke and Jeeves pattern search, (3) Powells method of conjugate directions and (4) Fletcher-Reeves conjugate gradient method. The following subsections give brief description of these methods. More details are available in standard optimization texts such as Belegundu and Chadrupatla (1999).

#### 3.2.1 *Nelder-Meade Simplex method (NMS)*

The simplex method of Nelder and Meade is a popular local direct search method for unconstrained nonlinear optimization problems. Readers should not confuse this with the popular Linear Programming simplex method applied for linear constrained optimization problems. The Nelder-Meade simplex method is based on moving and resizing a multidimensional polygon (simplex) along a downhill direction until a local minimum is encountered. The simplex is formed by a set of *n*+1 points in an n-dimensional decision space. For example, if we have 4 unknown variables in our inverse problem then we have a 5-point simplex. In the N-M simplex method we traverse the decision space by resizing and/or moving the simplex until a local optimum is found (in this case the simplex converges to a single point). The basic operations in the method, reflection, expansion, and contraction along with the function values at each simplex point dictate how we resize or move the simplex.

The initial simplex of $n$+1 corners is constructed from the initial guess by perturbing each point by a fixed amount. The points with highest, second highest and the least function values are selected. The point with largest function value is excluded and mean of the points is computed. A new point is obtained by reflecting about the mean and the forward function is evaluated at the reflected point. If the reflection is successful then the point is further expanded. If the expansion step fails then the point is contracted. If both expansion and contraction stages fail then the previously reflected point is accepted. If reflection, expansion and contraction stages fail, then a scaling operation is used to scale the point with the least function value, which shrinks the simplex. The operations described above can be represented in notation form. Let $X_h$, $X_s$, and $X_l$ be the points with highest, second highest and lowest function value points and $X_b$ mean of the points excluding highest, given as $X_b = \frac{1}{n}\sum_{\substack{i=1 \\ i\neq h}}^{n+1} X_i$. The new points during reflection, expansion, contraction and scaling stages are calculated as follows,

$$X_r = X_b + r(X_b - X_h) \leftarrow \textit{Reflection}$$

$$X_e = X_b + e(X_r - X_b) \leftarrow \textit{Expansion}$$

$$X_c = X_b + c(X_b - X_h) \leftarrow \textit{Contraction}$$

$$X_i = X_l + s(X_i - X_l) \leftarrow \textit{Scaling}$$

The coefficients *r*, *e*, *c* and *s* are assumed as 1, 1, 0.5 and 0.5 respectively. The steps are repeated until the convergence or stopping criteria is satisfied.

The best performing GA solution is passed as the starting point for the simplex. A stopping criterion based on the maximum number of cycles or no improvement in the objective function value for some specified number of cycles is used. Note the procedure described here is for minimization. More details regarding this method can be found in several texts including Belegundu and Chandrupatla (1999), Borse (1997), Press et al. (1996).

### 3.2.2 *Hooke - Jeeves pattern search method (HKJ)*

Hooke and Jeeves method is a simple yet powerful exploratory pattern search method that can be applied to discrete and continuous optimization problems. The method has two basic steps: (i) explore the neighborhood of the current point and establish a pattern to move, (ii) move to a new point using the established pattern. The exploratory step consists of sequentially perturbing (positively and negatively) the current solution vector (starting with an initial guess) in each direction by a fixed amount (step size) such that an improvement is found in the solution. If there is no improvement after perturbations in all directions are completed, then the exploration is conducted with a reduced step size. The step size is progressively reduced until an improvement is found or it reduces below a prescribed tolerance (say 1e-6) in which case the algorithm is terminated. When an improvement is found, a pattern direction vector is evaluated by taking the difference between the improved solution vector and the old solution vector. In the second step, the new solution is found by extrapolating the old solution along the pattern direction.

A step size is chosen and exploration is started from the given starting point. Assume $X_{i0}$ is the initial starting point with *n* decision variables, where $e_i$ is the unit vector (*n* * 1) along direction *i* and *s* is the step size. The exploration and pattern step can be calculated as follows,

Exploration

$X_{ij} = X_{i0}$, for $i \neq j$ and $i, j = 1,..,N$

$X_{ij} = X_{i0} \pm e_i s$, *for* $i = j$ and $i, j = 1,..,N$

Pattern search

$X_{iJ} = 2X_{iJ} - X_{i0}$, $i = 1,...,N$ and $J = j$ (exploration succesful)

A correct step size is important for good performance of the algorithm. Usually a value in the range of 0.05 to 1.0 is selected. A stopping criterion based on the maximum number of iterations or the reduction in step size value below a prescribed tolerance (1e-3) is used. A complete description of this algorithm can be found in Belegundu and Chandrupatla (1999).

### 3.2.3 *Powell's Method of conjugate directions (PWL)*

Powell's method of conjugate directions carries out minimization along successive directions that are conjugate with respect to all previous directions. Powell developed an idea of computing conjugate directions without using derivatives [Powell (1964)]. It requires *N* single variable minimizations per iteration and sets up a new conjugate direction at the end of each iteration. The procedure for computing the conjugate direction set is as follows [Press et al. 1996],

*Initialize the set of directions* $U_i$ *to the basis vectors,* $U_i = e_i$ for $i = 1, ...., N$

*Repeat the following sequence of steps until function value stops decreasing:*

*Save starting position as* $P_0$

*For* $i = 1, \cdots, N$ *move* $P_{i-1}$ *to the minimum along direction* $U_i$ *and call this point* $P_i$

*For* $i = 1, \cdots, N$ *set* $U_{i+1} = U_i$

*Set* $U_N = P_N - P_0$

*Move* $P_N$ *to the minimum along direction* $U_N$ *and call this point* $P_0$

*Reinitialize the set of directions* $U_i$ *to the basis vector* $e_i$ *after N or N+1 iterations.*

After searching along all conjugate directions, a spacer step is introduced where a search is made from the current point along the coordinate directions. The process is repeated until the convergence or termination criterion is satisfied. A stopping criterion based on the maximum number of cycles or the minimization cycle that produces a small change in the variable values less than one-tenth of the required accuracy is used. The method is also sequential and like other methods its performance depends on the initial guess. A complete description of this algorithm can be found in Belegundu and Chandrupatla (1999).

### 3.2.3 *Fletcher - Reeves conjugate gradient method (CG)*

Fletcher and Reeves conjugate gradient method is a steepest descent method and can be considered as a conjugate directions method involving the use of the gradient of the function. By evaluating the gradients of the objective function, new conjugate directions are set up at the end of each iteration and hence, faster convergence can be achieved. The iterative procedure for the method is given below [Rao 1996]:

(1) Start with an arbitrary initial point $X_1$.

(2) Set the first search direction $S_1 = -\nabla f(X_1) = -\nabla f_1$.

(3) Find the point $X_2$ according to the relation $X_2 = X_1 + \lambda_1 S_1$

Where $\lambda_1$ is the optimal step length in the direction $S_1$. Set $i = 2$ and go to the next step.

(4) Find $\nabla f_i = \nabla f(X_i)$, and set

$$S_i = -\nabla f_i + \frac{|\nabla f_i|^2}{|\nabla f_{i-1}|^2} S_{i-1}$$

(5) Compute the optimum step length $\lambda_i$ in the direction $S_i$, and find the new point

$$X_{i+1} = X_i + \lambda_i S_i$$

(6) Test for the optimality of the point $X_{i+1}$. If $X_{i+1}$ is optimum , stop the process.

Otherwise, set the value of $i = i + 1$ and go to step 4.

The process is repeated until the convergence or a termination criterion is satisfied. A stopping criterion based on the maximum number of cycles or the minimization cycle produces a small change in the variable values less than one-tenth of the required accuracy is used. The gradient vector ($\nabla f$) is computed using central finite differences. Task parallelism equal to twice the number of decision variables is inherent in the algorithm and is exploited while doing the function evaluations for computing the gradient ($\nabla f$). It should be noted that, once the gradient vector is computed step (4) involves five simple dot products and 9 Saxpy's. The optimal step length '$\lambda_i$' is computed using golden section search. Like other methods its performance depends on the initial guess. A complete description of the algorithm can be found in Rao (1996).

# Chapter 4 - Parallel Implementation

Although solving inverse problems using optimization based methodologies offer great flexibility in problem formulation it can be computationally demanding. Parallel computing can be used to alleviate this problem. The solution for the inverse problem is obtained by repeatedly solving the forward problem, and if the solution process of the forward problem itself is computationally intensive, then it becomes extremely important to have an efficient parallel implementation. An efficient FEM based parallel groundwater simulator suite PGREM3D [Mahinthakumar 1999], is used for forward function evaluations in this investigation. More details regarding the simulator can be found in the following articles: Saied and Mahinthakumar 1998 (flow simulator) and Mahinthakumar and Saied 1999 and 2002 (transport simulator).

The two most popular message-passing environments are parallel virtual machine (PVM) and message-passing interface (MPI). Our parallel implementation utilizes the latter. MPI is a popular portable, standard, parallel programming library and supports different languages like Fortran77, Fortran90, C, C++ and Java. It is widely supported on most parallel supercomputing architectures and distributed computing environments. MPI provides a convenient mechanism for modularizing parallelism through the use of "communicators". A communicator is a handle for facilitating communication among a specific group of processors. It enables message passing between processors and provides mechanisms for subdividing existing groups into new partitions and to send messages within and in between new partitions. Since groups can be further subdivided by the use of communicators, multiple hierarchical levels of parallelism leading to massive parallelism can be achieved through the simultaneous exploitation of coarse-grained parallelism in the optimizer and fine-grained parallelism in the function evaluator.

For example, the coarse-grained task parallelism in GA is generally restricted to the number of individuals in a population. However, if the forward solution process is already parallelized, then the amount of parallelism available is not just additive but also multiplicative. For example, if 10 function evaluations are performed concurrently using 1 processor for each evaluation then only 10 processors can be used; but if 5 processors

are used for each evaluation then 50 processors can be used. In this scenario parallelism is exploited at a finer level for each forward function evaluation and inherently two levels of parallelism exist: one at the GA population level and the other at the function evaluation step. In addition to increased parallelism, such an implementation can lead to increased flexibility (ability to solve large forward problems) and in some cases, improved performance due to cache effects [Mahinthakumar and Gwo, 1999: Sayeed and Mahinthakumar, 2002]. The sections that follow describe the parallel implementation of the hybrid optimization framework for a single supercomputer and its extension to the grid environment.

### 4.1 Coupled FEM-GA-LS implementation

In this implementation, the FEM transport and optimization modules are combined in to a single executable. This is in contrast to a previous implementation [Mahinthakumar and Gwo 1999], which used two separate executables for the GA and FEM modules, and employed the PVM library [Parallel Virtual Machine; Geist et al. 1994] for communication between the GA and FEM executables. While less modular, combining the optimization and FEM modules into a single executable has two main advantages: (i) the more portable MPI library can be exclusively used, (ii) the costly startup overhead for spawning each FEM simulation can be eliminated. The current implementation uses a three-tier communication hierarchy, and uses communicators at specific levels for reading input files and broadcasting it to other processors at that level, thereby reducing costly I/O time. A self-scheduling algorithm keeps all the server processors in a group busy; however at the end of each generation/cycle the processors are synchronized. A restart option is also available for the GA to restart its operations from where it stopped.

An important feature of the present implementation is the exclusive use of the MPI library. Use of MPI provides improved portability to a wide range of parallel architectures. The use of “communicators” in MPI provides a convenient way to couple the GA/LS and FEM modules. In MPI, communicators can be assigned to any group of processes. Communicator serves as a handle to that group. Within each group, each

processor has its own local process id (also called “rank”). Ranks range from 0 to *n*-1 within each group, where *n* is the total number of processes in the group. By default, all processes are assigned to the “world group” and the group handle is the “MPI world communicator”. Any number of subgroups can be created from the world group, and additional groups can be created from each subgroup. Once a subgroup is created, each processor will have a local process id and all local communication within that group can be handled using the “group communicator”. By hierarchically creating subgroups we can elegantly manage multi-level communication. More details on the use of MPI groups and communicators can be found in any MPI book (e.g., Gropp et al. 1999; Snir et al. 1996; Quinn 2004).

The algorithm/implementation has three levels with all the processors at the top level (level 0) belonging to the world group. At the next level (level1) the processors are divided in to several groups depending on the number of concurrent multi-type or multi-start GA/LS searches. This information about the number of multi-start/multi-type searches, the total number of processors assigned to each search, the number of processors assigned to each function evaluation, and the corresponding methods for each search is provided by the user. One processor in each group exclusively performs the computationally trivial GA/LS operations and serves as the manager or client processor. Note that in our discussion the words “processes” and “processors” are interchangeable since we always associate 1 process with 1 processor (or CPU). Each objective function evaluation (forward FEM transport simulation) can be assigned to a single process or to multiple processes of a server subgroup. The subgroup here refers to the number of processors used for each forward function evaluation. When a single process is used then each subgroup has just one process and the number of server subgroups equals one less than the total number of processors in that group. When multiple processes are used, the number of server subgroups equals the total number of server group processes divided by the number of processes per subgroup.

The processes and groups corresponding to our implementation are schematically shown in figures 4.1 and 4.2. Figure 4.1 depicts the structure of our GA/LS hybrid optimization framework. For GA the task parallelism is limited to the number of

individuals in the GA population. The best performing GA individual or a set of modeling to generate alternatives (MGA) (see Appendix - B) can be passed as initial starting guesses to the local searches. Multiple local searches, either same or different using same/different initial guesses can run concurrently. The multilevel communication hierarchy for a group (as several groups performing different GA or LS operations can be created concurrently) is shown in figure 4.2. The communication can be carried out at three levels between the client process and server processes. Let's assume we have a total of *N* processors for our coupled GA-FEM simulation. At first, all the *N* processes are first assigned to the world group with the default communicator MPI_COMM_WORLD (level-0). The processes are then divided into *n* groups with specified number of processors per group. The processes belonging to a group have their own communicator (level 1) and one of the processes will be the client and the rest are server processors. The group server processes are then further divided into *P* subgroups with each subgroup having *p* processes. Each of these *P* subgroups are assigned a server subgroup communicator (level-2). Each server subgroup will perform one transport simulation at a time. In each level, local process id numbers (or ranks) will be assigned to each process. Since the basic input file is the same for all the server subgroups performing the transport simulations, only one process (in our case, the process with rank 0) at level-1 will need to read the input file. Once read, the input data can be broadcast to all the other server processes using the level-1 group communicator. This mechanism avoids the need for each server process to read the input file and thus preventing I/O conflicts and also possibly saving on costly I/O time. All local communication within each transport simulation is handled using the level-2 communicator.

The manager process first sends the string representing the unknowns (decision variables) to the server processes. The server subgroups complete the transport simulation and return the RSE value (objective). In each subgroup, only the processor with rank 0 or "group leader" communicates with the group manager processor. When multiple processes are used in the server subgroups to do the forward transport run, the process with local server subgroup rank 0 will receive the individual (chromosome string) from the manager and does a broadcast within its subgroup. The subgroup processes will

perform the transport run and the process with local rank 0 in this subgroup will communicate back the RSE value to the manager using the level 1 group communicator.

Initially, all the server processes in the server subgroups with local rank zero will receive an individual (chromosome string) from the client, after which the individuals are assigned dynamically to whichever server subgroup returns the RSE value first to the client. The client process keeps sending the individuals in a population to server subgroup processes until all individuals in a population of a generation is completed. However, the client process needs to synchronize the processors at the end of each generation. The dynamic dispatching of the individuals by the client process will help keep all the processes busy in a homogeneous or heterogeneous grid-computing environment, where the processors with different speeds and architectures are used. This dynamic task scheduling policy helps achieve load balancing especially if we have a small number of processors and a large population size. Typically we use a population size of 128 or 256 and 65 or 129 processors on IBM SP, with one process (client) dedicated to do the GA computations. The load balance results obtained are presented in chapter 6.

### 4.1.1 *Hybrid GA-LS-FEM implementation*

As mentioned earlier the LS methods are generally used for local fine tuning. Once the GA has exhausted in its search by not improving the function value or by reaching the pre-specified number of generations, the best solution is passed as the initial guess to the local search methods. The search is then carried on by these methods. The LS methods are basically sequential in nature except for NMS and CG methods. In NMS method limited amount of task parallelism (up to the number of decision variables) is available and exploited during some of the iterations when multiple points of a simplex need to be computed each requiring a function evaluation. In the CG method when gradients are computed using central finite-differencing, task parallelism equal to twice the number of decision variables is available and is exploited. The sequential bottleneck in local searches can be alleviated by either increasing the fine-grained parallelism or by performing simultaneous multi-type or multi-start local searches. Similar to GA the LS methods send the decision variable string to the server processes for forward function

evaluation and the function value is returned back to the client process. The creation of groups, subgroups and the communication operations performed are the same as described before except that now the optimizer is a LS method instead of GA. Normally we restrict the maximum number of server subgroups to the number of decision variables for the simplex method. For CG method the maximum number of processors is restricted to twice the number of decision variables, excluding one for the client. Dynamic task scheduling is also implemented in the NMS and CG methods. Different convergence and termination criteria can be used to stop the simulation. The LS algorithm is terminated when: (a) it has completed the total number of cycles/iterations, (b) there is very small improvement for five iterations and (c) no improvement in the search direction.

### 4.2 Grid implementation

Grid-computing environments are an emerging trend in parallel computing resources that typically consist of a collection of geographically distributed heterogeneous supercomputer resources (e.g., the NSF's proposed new distributed terascale facilty[1] (DTF)). Parallel implementations for these environments are inherently multilevel and obtaining efficient mapping of work to processors can be extremely challenging. Extension of our previous implementation to the grid can be accomplished by using "grid-enabled" version of MPI libraries. Using the Globus toolkit and grid-enabled MPI (MPICH-G2 or VMI2-MPICH), required number of resources can be requested from multiple supercomputers. Grid-enabled MPI is a special version of MPI suitable for computational grids and is based on the "Globus" meta-computing toolkit (MPICH-G2) or the virtual machine interface[2] (VMI). MPICH-G2 is a Globus flavor of MPICH using services from the Globus toolkit (such as job startup, security etc.,). VMI2-MPICH uses middleware communication layer VMI with MPICH. MPI applications can be run on multiple machines potentially of different architectures using grid enabled MPI libraries. These libraries automatically convert data in messages sent between machines of different architectures and support multiple underlying communication protocols.

[1] http://www.nsf.gov/od/lpa/news/press/01/pr0167.htm
[2] http://vmi.ncsa.uiuc.edu/

Initially all the processors requested on different machines (supercomputers) belong to the world group and have MPI_COMM_WORLD as the global communicator at top level (level 0). Similar to the implementation described above multiple groups can be created with their own communicators (level 1). One of the processors in each of the groups will be the client and the rest server processors. The server processors of the group (level 1) are then divided in to server subgroups (level 2). All the processes of a server subgroup (level 2) performing fine-grained FEM computations should be confined to a single supercomputer to prevent costly latency overheads. Of course the challenge here is that at the top level all the processes may have a global ranking independent of the location of processes on the supercomputers, i.e., the MPI library may assign processor ranks that may not be in a sequential order. Therefore the server processes with different global rankings on a supercomputer have to be identified and regrouped into subgroups that are local to each machine.

To address this we use the MPI call "MPI_Get_processor_name" to get the processor's name and then check its locality during subgroup formation. Once these multilevel groups and communicators are created for the heterogeneous or homogeneous grid environment, the communications or computations follow the same approach described for a single supercomputer. We investigate parallel performance and other issues related to the grid-computing environment for this application in chapter 6. A discussion of grid and grid applications are available in several recent papers [Waldrop 2002, Johnston 2002, Abramson et al 2002, Natrajan et al].

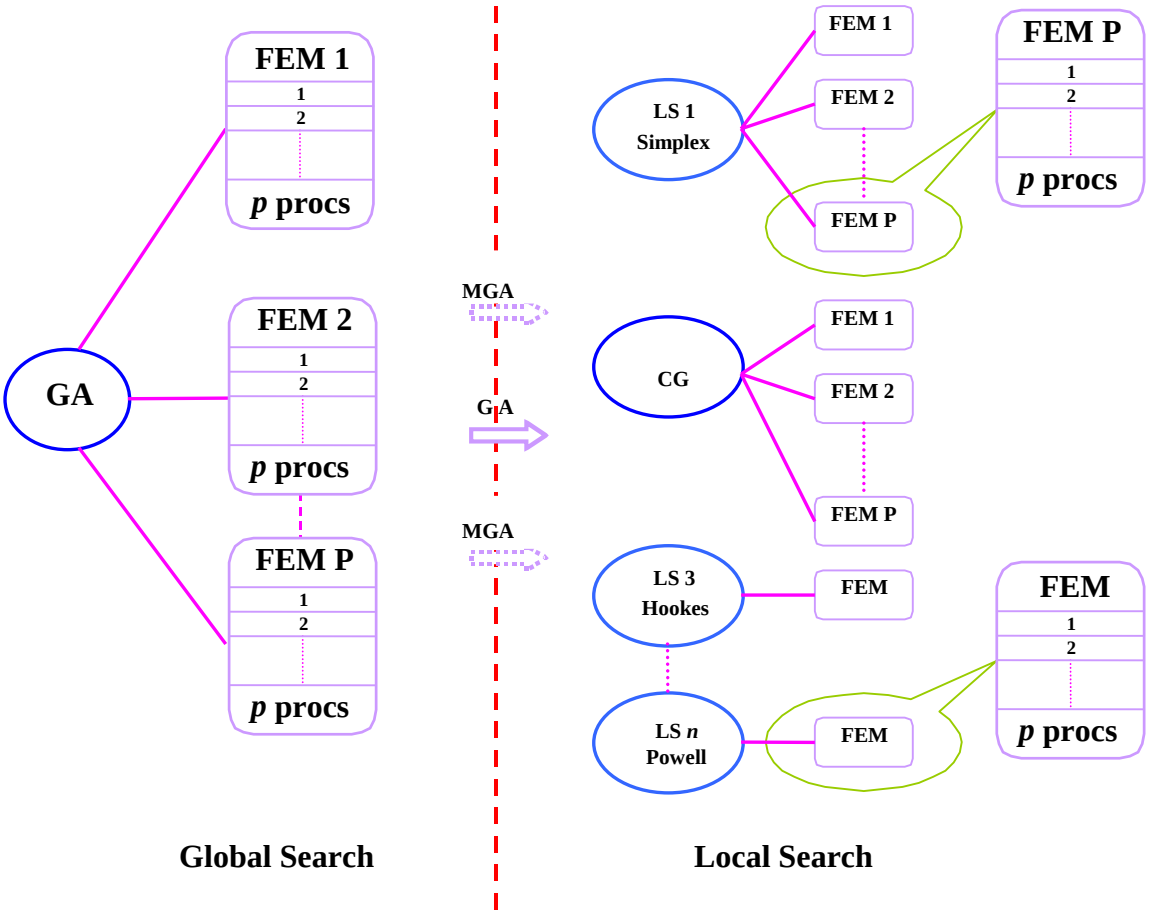



**Figure 4.1 Schematic layout of parallel hybrid GA-LS-FEM optimization framework. The GA solution or the MGA alternatives are passed as initial starting guess to local search methods. The GA has *P* tasks (individuals) evaluating using *p* processors for each function evaluation. The local search can be performed with *n* different/same methods using same or different initial starting points.**

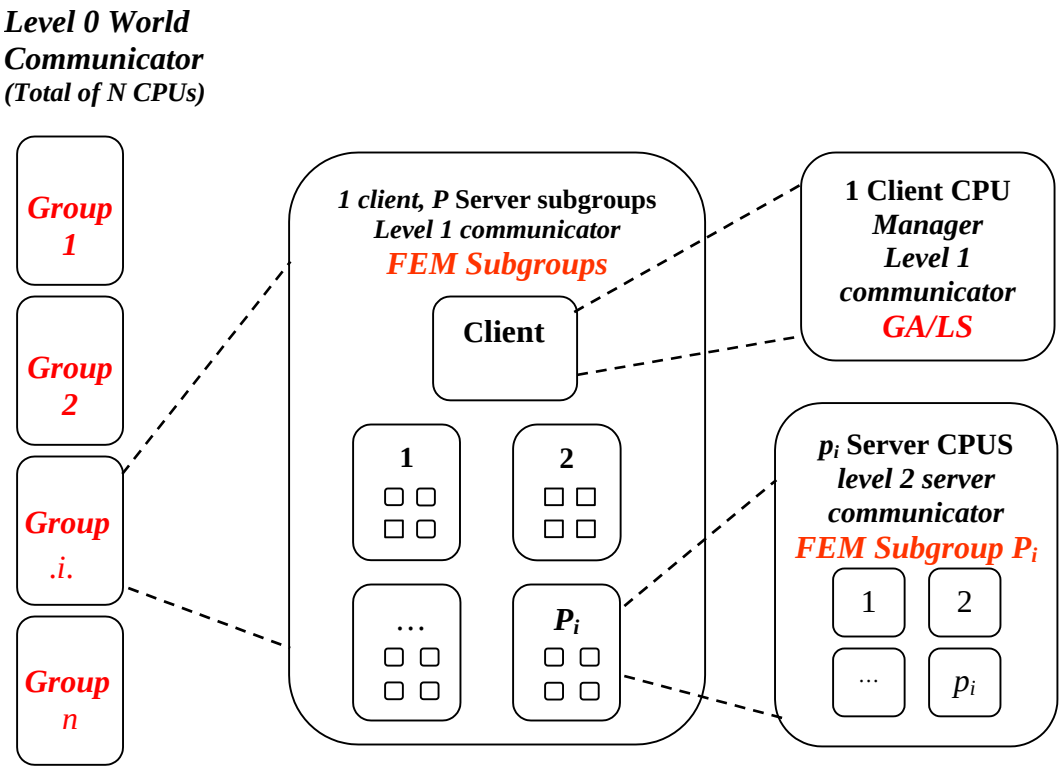



**Figure 4.2 Three levels of MPI communicator hierarchy with multiple groups (*n*) performing GA/LS operations, and each group using different number of processors in a group ($P_i.p_i$ + 1). $P_i$ is the number of server subgroups for group *i* using $p_i$ processors for each FEM forward function evaluation. One processor in each group is dedicated for GA or LS operations.**

# CHAPTER 5 - TESTING AND EVALUATION

The optimization framework has been tested for the following classes of subsurface characterization problems: biological activity zone identification, source identification and source release history reconstruction. For all test problems reference or "measured" data is synthetically generated and compared to the simulated observation data. Before presenting the test problems and results in the sections ahead, a brief explanation of the FEM transport simulator follows in the next section.

## 5.1 Description of the FEM transport simulator

The parallel transport simulator employed in the GA function evaluations solves the multi-component groundwater transport problem. The general system of equations describing transport of *nc* dissolved components undergoing reactions in saturated media is given by

$$\frac{\partial C_i}{\partial t} = \nabla \cdot (\mathbf{D} \cdot \nabla C_i) - \nabla \cdot (C_i \mathbf{v}) + \frac{q}{\theta}(C_i - C_{0i}) - R_i \qquad i = 1, 2, 3..., nc$$

where $\boldsymbol{v}$ is the 3x1 velocity field vector, $\boldsymbol{D}$ is the 3x3 dispersion tensor dependent on $\boldsymbol{v}$, and $C_i$ is the dissolved concentration of component *i*. The term $q(C_i\text{-}C_{0i})/\theta$ represents the source term with volumetric flux *q*, medium porosity $\theta$, and injected concentration $C_{0i}$ (e.g. from injection wells). $R_i$ is the rate of mass loss of component *i* due to sorption and bioremediation reactions and is the main coupling term for the system of equations. The term, $R_i$, may contain many terms and can be nonlinear. For example, if only bioremediation reactions are present then $R_i$ is given by

$$R_i = \mu_{\max} F_i X \prod_{\substack{j=1 \\ f_{ij} \neq 0}}^{j=nc} f_{ji} \left( \frac{C_j}{K_j + C_j} \right) \qquad i = 1, 2, 3, ..., nc$$

Where $F_i$ is the stoichiometric ratio, $X$ is the biomass concentration, $\mu_{max}$ is the maximum utilization rate, and $f_{ji}$ is a factor controlling component *j*'s contribution to component *i*'s

biodegradation process. If $f_{ji} = 0$ then component $j$ does not participate in component $i$'s biodegradation process.

The system of transport and reaction equations is discretized using the Galerkin finite element method (FEM) with 8-noded linear hexahedral elements. A logically rectangular grid structure is assumed but irregular geometries are supported using distorted elements. A Crank-Nicolson approximation (central finite-difference) is used for the time derivative terms. A lumped mass formulation [Huyakorn and Pinder 1983] is used for all time-derivative and non-derivative (zeroth spatial derivative) terms. The coupled non-linear system is solved using a modified form of the Sequential Iterative Algorithm (SIA). Several Krylov subspace iterative solvers are implemented in the code for the matrix solution [Mahinthakumar et al. 1997]. In this research BiCGSTAB solver is chosen for the simulations, which performs reasonably well for most problems.

This transport simulator is parallelized using a two-dimensional domain decomposition (in the $x$ and $y$ directions) using explicit message passing (MPI library) to exchange information between these domains. The simulator has been tested extensively for scalability and performance on a variety of parallel architectures. Details can be found elsewhere [Mahinthakumar and Saied 1999, Mahinthakumar and Saied 2002].

### 5.2 Biological activity zone identification problems

Identifying zones of biological activity is critical to the efficacy of bioremediation measures. Bio-stimulants such as dissolved oxygen and methane are injected and the observed breakthroughs of methane are used to deduce BAZ. Bio-stimulants (methane, dissolved oxygen) are commonly injected into the subsurface to stimulate the growth of bacteria so that they can eventually degrade the contaminant to desired levels [Semprini and McCarty 1991].

The GA-FEM framework developed is used to solve the inverse problem of determining possible biological activity zones (BAZ) from the results of a bio-stimulation experiment. In these experiments indigenous methanotropic bacteria are stimulated by continuous periodic injection of dissolved oxygen and methane (see Figure 5.1). The

observed breakthroughs of methane are used to deduce possible biological activity zones in the subsurface.

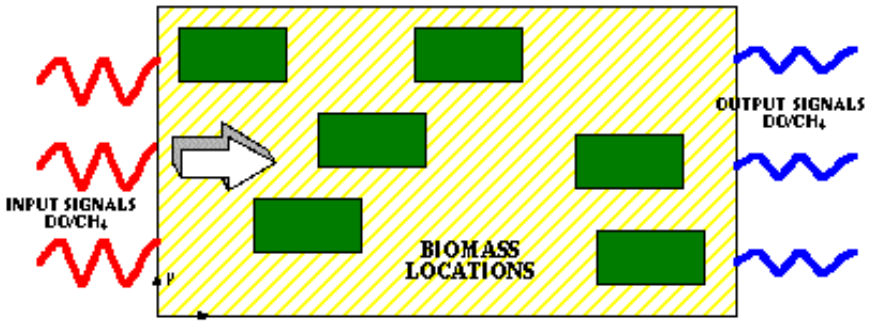



**Figure 5.1 Problem setup for the biological activity zone identification problem.**

The problem domain is divided in to several distinct zones (for e.g. 36) resulting in the same number of binary or integer bits for GA encoding. The binary representation encodes the zone location and activity, with 0 for inactive and 1 for active. Integer encoding additionally indicates the activity (concentration) levels. This problem is inherently discrete and therefore BGA/IGA is suitable for these problems. Since the standalone BGA/IGA performed reasonably well and because local searches are not amenable for discrete representation the hybrid approach was not investigated. GA performance is investigated for problems of varying complexity (e.g., for identifying three and ten BAZs) using different zonal encoding and GA operators.

### 5.2.1 *Description of test problems*

We chose a moderate size problem with a grid resolution of 51 x 31 x 11 (17,391 finite element nodes). The grid spacing is fixed at 0.2 m in each direction leading to a problem domain of 10m x 6m x 2m. Stimulated methane concentrations are observed at 9 downstream locations for a time period of 40 days (200 time steps). The observed methane concentrations are pre-calculated using an assumed zonal distribution. GA will attempt to find this distribution by minimizing the error between these pre-calculated values (observed or reference signals) and the computed values for each trial solution. Two types of zone identification problems are examined, three zone identification and ten zone identification, and the results are reported in the following sections.

### 5.2.2 *GA Encoding scheme*

The 10 x 6 x 2 m domain is divided into 36 rectangular zones (4 x 3 x 3 decomposition) numbered in the two different encoding schemes as shown in figure 5.2. The length of binary/integer bit string for GA encoding is 36, one for each of the 36 zones. For the binary strings, the bits are either 0 or 1, and for integer strings, the bits are 0, 1, 2 or 3. If the bit is 0 for some zone location, then that zone is inactive and if it is not 0, then the zone is active. For integer encoding, the values denote an activity level of the zone; i.e., 0 – inactive, 1 – low, 2 – medium, and 3- high. These values are decoded into appropriate BAZ concentrations in the transport code. The locations of the bits in the chromosome string are encoded to correspond to the actual zone locations. As noted, for integer encoding, each string not only encodes the zone locations but also the corresponding biological activity level. We have studied two different encoding schemes. In encoding type A, a zone at $x = 1$, $y = 1$, $z = 2$ would correspond to a zone number 13, where as in encoding type B a zone at $x = 1$, $y = 1$, $z = 2$ would correspond to zone number 24. One would expect encoding Type B to perform better with single point crossover. This is because in encoding Type B, adjacent locations in the string correspond to adjacent locations in the real domain.

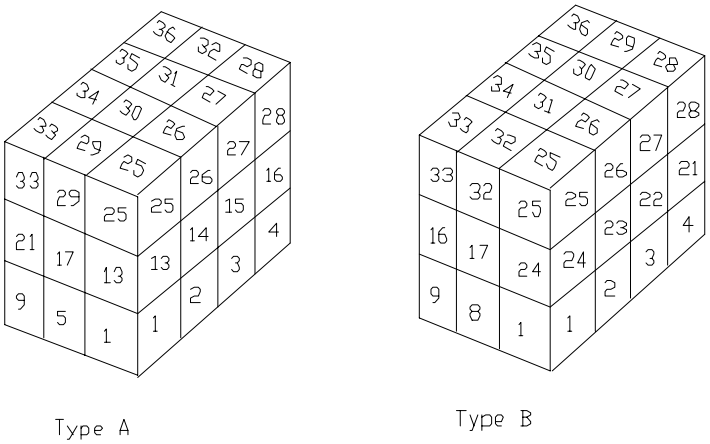



**Figure 5.2 Two types of zone encoding.**

### 5.2.3 *GA performance results*

The performance of GA for 3 and 10 zone identification problems is studied using binary and integer GA, and results are reported here. GA performance is measured in terms of convergence to the exact solution. For both three zone and ten zone cases, the initial population is generated using a probability of 0.5 whether a location is active or not (with activity levels in the case of integer encoded problems). Note that in real life, we may not actually know how many zones are active. If we had prior information regarding the number of zones that are active then we would have used a probability of 0.1 (3/36) for the three zones case and a probability of 0.3 (10/36) for the ten zones case. Thus an unbiased initial population generation is more realistic. Several random seeds were tried out for generating the initial population. Convergence rate varied slightly for different random seeds and the results corresponding to the median performing seed are reported below. In all cases, a crossover probability of 0.4 and an initial mutation probability of 0.01 are used. These values were chosen based on a few trial runs.

*Three zone problem*

For the reference case three arbitrary zones were chosen with numbers, 17, 20 and 34. For encoding type A, these numbers would correspond to ($x$,$y$,$z$) coordinates of (1,2,2), (4,2,2), and (2,3,3) respectively and for encoding type B, these numbers would correspond to (2,2,2), (4,2,2), and (2,3,3). A fixed population size of 256 is used for all cases based on the approximate thumb rule of 3 to 4 times the chromosome string length. A fixed initial random seed is used for all cases. The GA convergence plots are shown in figures 5.3 and 5.4 for both encoding schemes using uniform and simple crossover respectively. The figures show the average RSE values for the population after each generation. The results show that for both simple and uniform crossover, encoding type B performed better. Encoding B's advantage, however, is more pronounced for simple crossover. This result is expected as discussed earlier.

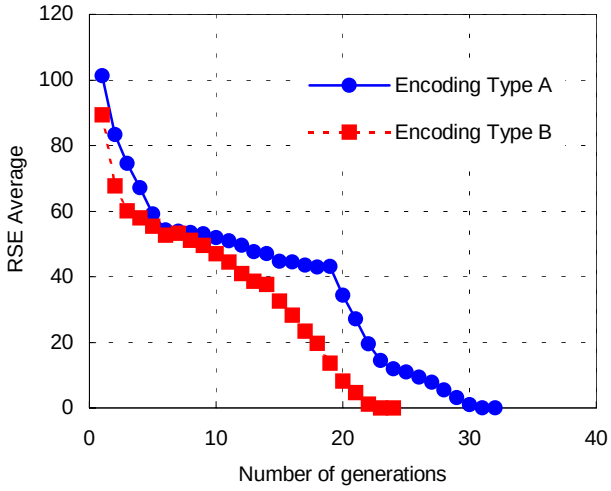



**Figure 5.3 GA convergence history for 3-zone problem using uniform crossover.**

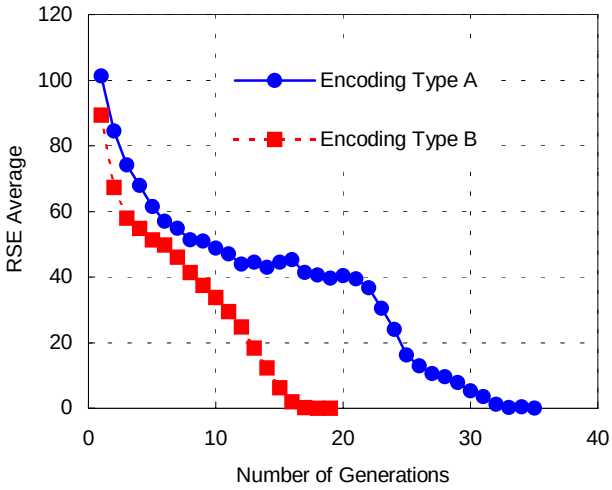



**Figure 5.4 GA convergence history for 3-zone problem using simple crossover**

*Ten zone problem*

This is a more difficult problem than the three-zone identification problem for the GA to solve, as it is trying to identify ten active sites out of a possible 36. Doing elementary combinatorial analysis (Spiegel 1975) shows that there are ${}_{36}P_{10,26} = 36!/(10! \cdot 26!) = 2.5 \times 10^8$ possibilities! The arbitrarily chosen BAZ in the reference case are zone numbers 1, 2, 5, 7, 10, 14, 18, 19, 25 and 32. The figures 5.5 and 5.6 show the performance of GA for this problem using simple and uniform crossover respectively. The same GA parameters as in the three zones case are used.

Although it is not entirely clear as to which encoding scheme or crossover strategy is better for this problem, from the initial convergence behavior we see that encoding type B and simple crossover perform better. We note here that the convergence of encoding type A in figure 5.6 around the $70^{th}$ generation seems to be a random phenomenon as this behavior could not be consistently reproduced. Obviously, the GA convergence is much better for the 3-zone identification problem than this 10-zone problem (compare figures 5.3 and 5.4 with 5.5 and 5.6). For the 3 BAZ problem the exact solution is obtained as indicated by the zero RSE value, where as the ten zone BAZ problem is more complicated and only nine out of ten zones were correctly identified for encoding type B and uniform crossover strategy.

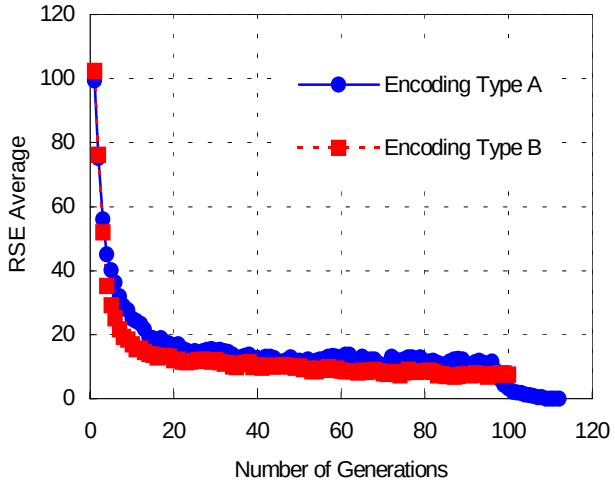



**Figure 5.5 GA convergence history for 10-zone problem using uniform crossover.**

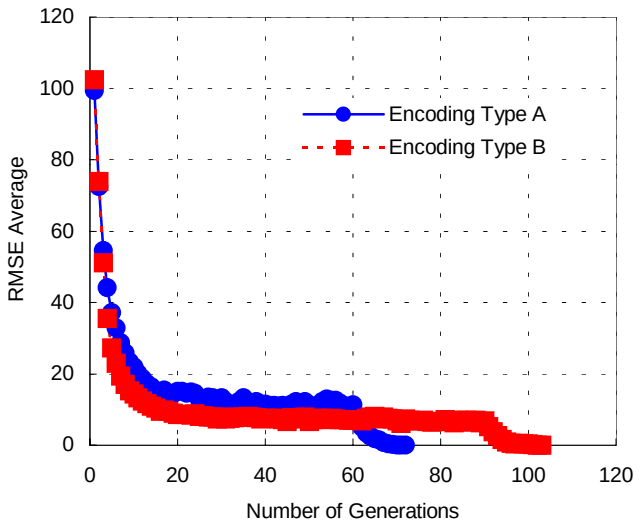



**Figure 5.6 GA convergence history for 10-zone problem using simple crossover.**

### 5.2.4 *Integer encoding problem*

As noted earlier, in this problem we are trying to determine the locations as well as the activity level of each zone. This is a more difficult problem than the previous BAZ problems for the GA to solve, as it is trying to identify ten active sites out of a possible 36

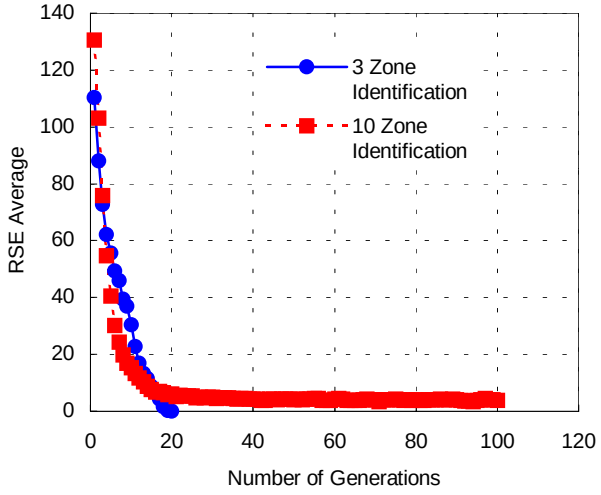



**Figure 5.7 GA convergence history for integer encoding problem. Encoding type B and simple crossover are used.**

and also their activity level (4 levels). Doing elementary combinatorial analysis [Spiegel 1975] shows that there are 4 times ${}_{36}P_{10,26}$ = 4. 36!/(10! . 26!) = 1.0 x $10^9$ possibilities! The arbitrarily chosen BAZ in the reference case are zone numbers 1, 2, 5, 7, 10, 14, 18, 19, 25, and 32. The convergence results are shown in figure 5.7 for both the 3 zone and 10 zone problems. Obviously, the 10-zone case is a difficult problem and thus we see that the GA did not find the exact solution. Further examination of the solution found by GA indicated that it had correctly identified 9 out of 10 zones in this case. Given that GA is a global search technique, this is very good!

### 5.3 Source identification problems

Identifying contaminant sources (locations and concentrations) is important in the design of efficient remediation strategies and identifying responsible parties in a contamination incident. This is an inverse problem and the solution has to be generally inferred from sparsely available concentration measurements. A hybrid GA-LS approach is used for solving, as preliminary tests using standalone GA or LS failed to perform well. A number of observation wells uniformly distributed at the mid and downstream vertical sections are situated as shown in figure 5.8 for sampling the synthetically generated plume. The unknown decision variables are the coordinates of the zones and their concentration values. For example, for a single source problem the 7 decision variables are the 6 coordinate values [($x_1$, $y_1$, $z_1$), ($x_2$, $y_2$, $z_2$)] and the initial concentration $C_0$. A more detailed description is provided in section 5.3.1.

*Satisfying the continuity requirement*

The inverse problem, as posed above, is a mixed discrete-continuous optimization problem since the decision variables are mixed with discrete integer valued grid coordinates $x_1$, $y_1$, $z_1$, $x_2$, $y_2$, $z_2$ and a continuous real valued initial concentration $C_0$. While discrete decision variables are not a problem with GA, most of the local search methods employed here (with the exception of Hooke-Jeeves (HKJ)) requires that the decision variables be continuous with respect to the objective function (continuity requirement). This restriction called for a modification of the forward problem such that a small real-valued change in the coordinates is guaranteed to produce a corresponding change in

objective function. This was accomplished by transforming the problem by extrapolating the concentration values to the nearest exterior grid points of the source zone. This extrapolation procedure is shown in figure 5.9 for a 2D problem. While the use of integer

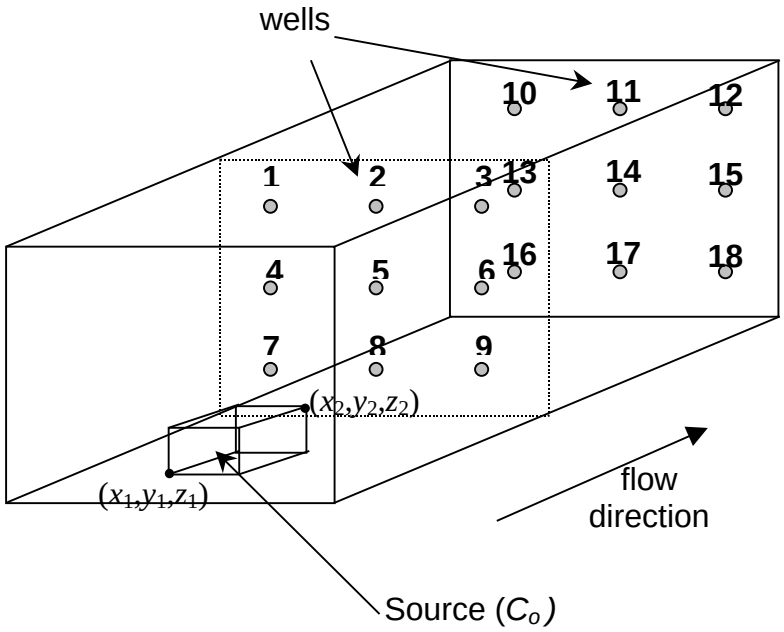



**Figure 5.8 3D domain with a single source and observation well locations.**

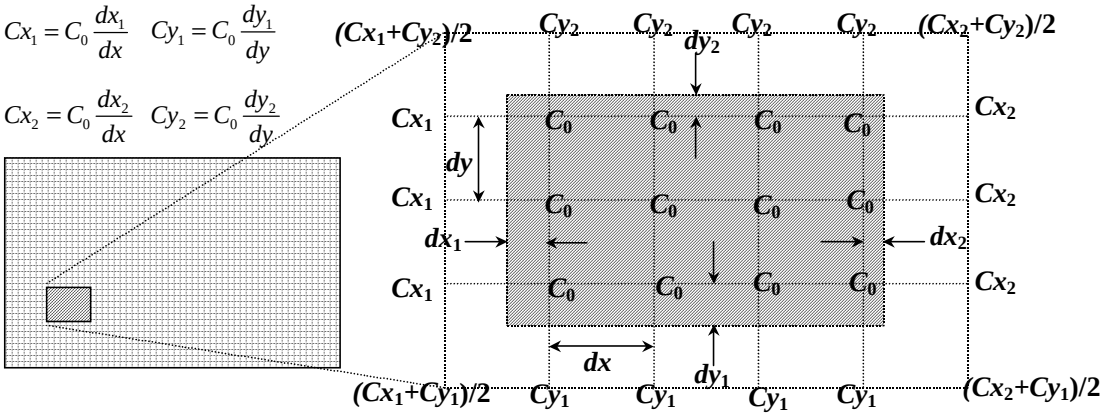



**Figure 5.9 Concentration extrapolation scheme for a 2D problem. A similar approach is applied to 3D cases.**

valued grid points to represent the boundaries of the source is somewhat artificial for this particular problem, there may be other inverse problems where integer valued representation may be a necessity (e.g., zone identification problems).

### 5.3.1 *Description of test problems*

The problem solves 3D pollutant source location from measured concentrations downstream of the source. The problem domain and grid resolution are the same as that of the BAZ identification problems. The 3D problem uses a heterogeneous flow field and the steady state flow field is generated using a parallel groundwater flow solver [Saied and Mahinthakumar 1998]. The randomly heterogeneous hydraulic conductivity field for the flow solver is generated using a 3-Dimensional turning bands code. The turning bands code is a parallelized version of the original code developed by Andy Thompson [Thompson 1987]. The log conductivity field uses a mean of zero and a variance of 1. The source release is observed for 20 days (1000 time steps). The assumed source corresponds to coordinates $x_1$=8, $y_1$=9, $z_1$=3, $x_2$=16, $y_2$=15 and $z_2$=7 and having an initial concentration of 72 mg/L. A total of 18 observation wells with 9 each uniformly distributed at the mid and downstream vertical sections are utilized. The observations are recorded once every 10 time steps. Therefore, a total of 1800 observations are used corresponding to 18 wells and 100 time periods. The problem was made continuous for local search methods by using the weighting approach described earlier (Figure 5.9). By using a weighting approach any small change in the coordinates is guaranteed to produce a change in the objective function value. The objective function is the root square error (RSE) between the observed and calculated values

$$RSE = \sqrt{\sum_{i=1}^{n} (C_i^{obs} - C_i^{calc})^2}$$

where $C_i^{obs}$ is the observed concentration and $C_i^{calc}$ is the calculated concentration at the observation points, *i* the observation number and n the total number of observations.

### 5.3.2 *Hybrid GA-LS performance results*

An investigation on 2D problems showed that stand-alone GA is not adequate for these problems and a hybrid approach is warranted [Mahinthakumar and Sayeed 2003]. A more realistic scenario is used for the 3D case tested here with heterogeneous flow field and an increased number of observation points (18 points). Hybrid GA-LS approaches based on both BGA and RGA were tested on this problem. The GA portion was run for 35 generations and the best solution was further fine-tuned by the local searches. The BGA used the following parameters: simple crossover, tournament selection with no replacement, population size = 128, probability of crossover=0.5, and probability of mutation=0.04. For RGA the parameter values are: population size =50, probability of crossover=0.7, probability of mutation=0.01, tournament selection with replacement, and non-uniform mutation. Instead of using the option to seamlessly switch from GA to LS, we terminate GA and then restart the local searches using the restart option. This enables us to test different local search methods from the same initial guess provided by GA without rerunning GA for each local search method. Multiple GA-LS runs were conducted and the results are reported for the average performing case. The results for the hybrid GA-LS approach for 3D source reconstruction problem are as shown in figures 5.10 (BGA) and 5.11 (RGA) for the problems without noise and figures 5.12 (BGA) and 5.13 (RGA) for problems with noisy data. The noisy data is obtained by adding a random white noise of ± 10% to the synthetically generated observation data. Figures 5.10–5.13 show that the Nelder-Meade simplex method (NMS) performed well initially but as the size of the simplex starts to reduce, the convergence rate dropped. The same can be said about Powell's method (PWL). Table 5.1 shows that even though all the hybrid approaches are able to find a reasonably good approximation to the solution, the Hooke and Jeeves method (HKJ) required the least number of function evaluations to converge. Also, RSE values do not always correspond to the error in the solution indicating a non-uniqueness problem. For example the BGA has higher RSE value (41.27) than RGA (21.89), but has a lower % error in solution of 9.7 compared to 11.4 for RGA.

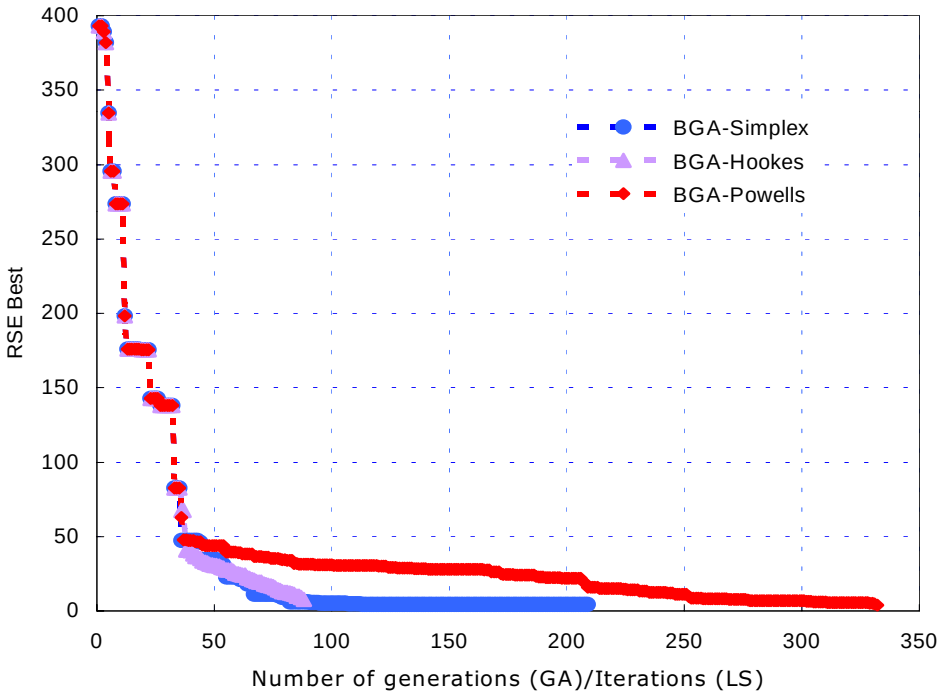



**Figure 5.10 Convergence history of Hybrid BGA-LS approach for 3D source identification problem with no noise. Iterations for Hooke's and Powell's method refer to every reduction in RSE value with forward function evaluation.**

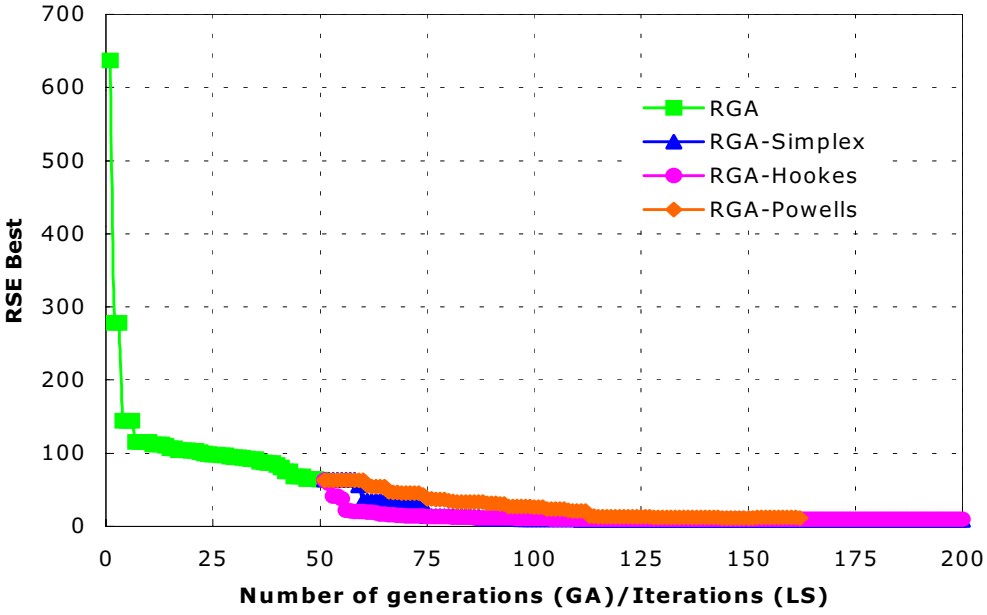



**Figure 5.11 Convergence history of Hybrid BGA-LS approach for 3D source identification problem with no noise. Iterations for Hooke's and Powell's method refer to every reduction in RSE value with forward function evaluation.**

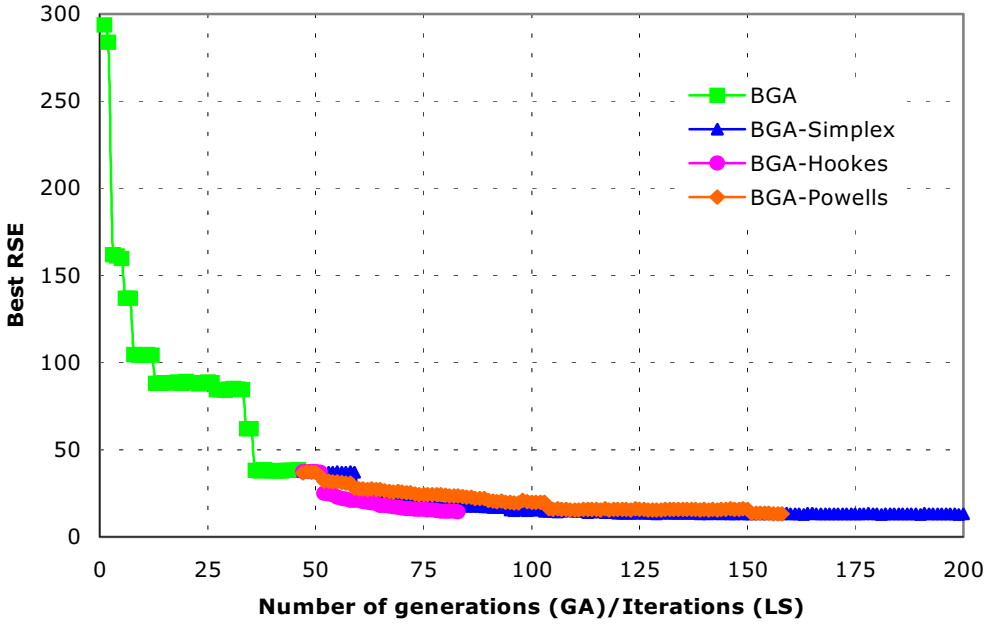



**Figure 5.12 Convergence history of Hybrid BGA-LS approach for 3D source identification problem with ±10% noise in observation data. Iterations for Hooke's and Powell's method refer to every reduction in RSE value with forward function evaluation.**

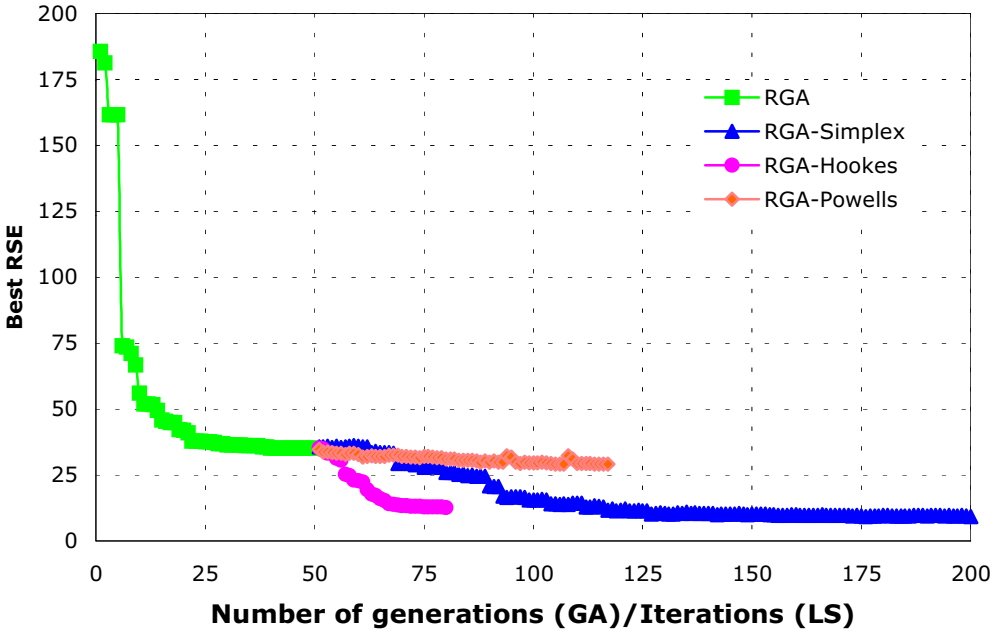



**Figure 5.13 Convergence history of Hybrid RGA-LS approach for 3D source identification problem with ±10% noise in observation data. Iterations for Hooke's and Powell's method refer to every reduction in RSE value with forward function evaluation.**

**Table 5.1 Error in solutions obtained using various methods for the 3D source identification problem with ±10% noise in observation data.**

The true solutions is $x_1$=8, $y_1$=9, $z_1$=3, $x_2$=16, $y_2$=15, $z_2$=7 and $C_0$=72 mg/L. RGA = real genetic algorithm, BGA=Binary, NMS (Nelder-Meade Simplex), HKJ (Hooke-Jeeves) and PWL (powells).

| Method | Number of Function Evaluations | Converged Solution ($x_1$, $y_1$, $z_1$, $x_2$, $y_2$, $z_2$, $C_0$) | RSE Value | Error in Solution (%) |
|---|---|---|---|---|
| BGA | 1660 | (7, 9, 3, 17, 17, 7, 46) | 41.27 | 9.7% |
| RGA | 2340 | (10, 9, 3, 15, 15, 8, 65) | 21.89 | 11.4% |
| BGA-NMS | 3086 | (8, 9, 3, 16, 16, 7, 50) | 15.86 | 8.0% |
| BGA-HKJ | 2035 | (8, 8, 3, 16, 16, 7, 49) | 14.92 | 9.4% |
| BGA-PWL | 2460 | (8, 9, 3, 16, 16, 7, 52) | 17.96 | 6.8% |
| RGA-NMS | 3771 | (8, 9, 3, 16, 15, 7, 68) | 8.85 | 0.8% |
| RGA-HKJ | 2724 | (9, 9, 2, 15, 15, 8, 66) | 14.58 | 7.4% |
| RGA-PWL | 3142 | (10, 9, 2, 15, 15, 8, 73) | 13.03 | 10.0% |

## 5.4 Source release history reconstruction problems

In most practical situations the locations of contaminant sources are known, but the time history of contaminant release in to the subsurface is not known. Reconstructing the temporal release history from available concentration measurements (i.e., solving the source release history reconstruction problem) is important in environmental forensics where potential polluters are identified so that financial and other liabilities can be imposed on the responsible parties. The hypothetical multiple sources are present at different locations in the domain (see figure 5.14). Concentration observations are available at the middle and downstream-end vertical cross-sections of the domain at various time intervals. A heterogeneous flow field generated by the flow code is used as described for the source identification problem.

For this problem, the possible contaminant source locations are known but the contaminant release concentrations corresponding to a specified number of time durations (time history of contaminant release at the source) are unknown resulting in the same number of unknown decision variables. For example, ten time durations corresponding to a single source will require ten decision variables. Typically, the desired resolution in

time is a single year. In this case, the release history (concentrations) is sought for 10 time durations corresponding to a 10-year release of the contaminant. Depending on the number of contaminant sources the number of decision variables is equal to the product of the number of time durations times the number of contaminant sources. For example, if the number of time durations is ten and five contaminant sources then the number of decision variables is 50 (unknowns).

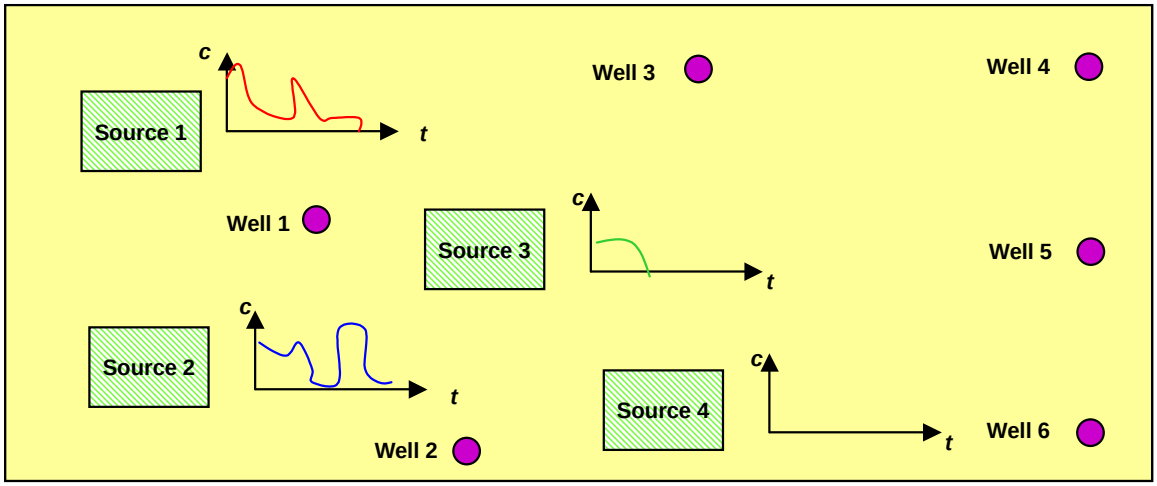



**Figure 5.14 Show's the 2D source release history problem.**

The problem domain, grid resolution and other specifications are same as for the source identification problem except that the potential source locations are known and the release history is sought for each source for the specified number of time durations. A hypothetical study of single and multiple (3 and 5) sources release is investigated. The sources are present upstream or downstream of the middle set of the observation wells. For these problems a total of 1800 observations (18 wells × 100 time durations) are recorded similar to the source identification problem. For these problems the locations are known and the contaminant release concentrations at each source over ten-year period are unknown resulting in 10, 30 or 50 decision variables for 1, 3 or 5 sources respectively.

### 5.4.1 *Hybrid GA-LS performance results for source release history reconstruction problem*

Convergence results are presented for single and multiple source problems with and without noise in the observations. For the cases with noise a random white noise of 10% is added to the observation data. The RGA is run with a population size of 100 and for 50 generations. The output profile frequency is set at 100. The main RGA parameters used are probability of crossover = 0.5, probability of mutation = .05. Of the individuals selected for crossover 40% undergo arithmetic crossover, 30% heuristic crossover, 20% uniform crossover and remaining 10% simple crossover. The mutation strategy selected for these runs is nonuniform mutation. The RGA was run for 50 generations using 101 processors on an IBM SP3 taking approximately 2 hrs 15 mins. The local searches are performed with fewer number of processors.

The performance of the hybrid approach for a single source problem is shown in Figures 5.15 (a) and 5.15 (b) for cases without and with noise. The values used for deltat and maxt parameters are 0.2 and 1000 respectively. The results show that the hybrid approach is a feasible approach for solving release history problems. Most of the LS approaches perfomed well, but the best is observed for CG method.

Results for the more complex multiple source release history (3 and 5 sources) problems are presented next. In both cases, one of the sources is assumed to be dummy; i.e., this source does not release any contaminants (zero concentration values). Dummy sources are not uncommon in practice as one or more of the potential pollution sources may not be leaching any contaminants. The parameters used for three-source release history reconstruction problem are same as the single source problem with the exception that deltat and maxt parameters are now 0.05 and 2500 respectively. The three-source results are shown in Figures 5.16 (a) and (b) for the cases without and with 10% white noise in observation data respectively. The performance of GA-LS varied and only the hybrid C.G approach perfomed well for both cases (with and without noise). However in most cases the dummy sources were clearly identified. It should be noted here that standalone GA or LS would have failed for multiple sources case.

For the five-source release history reconstruction problem population size is increased to 200 and the deltat and maxt parameters are set to 0.05 and 3500 respectively. Figures 5.17 (a), (b), (c) and 5.18 (a), (b), (c) show performance of the hybrid approach for the five-source problem for three LS methods (HKJ, PWL, and CG) for cases with and witout noise in observation data. The worst peformance is observed for simplex method and is not reported. Even the fine tuning of simplex method parameters such as step size didn't provide any significant benefit. These results are presented in a different format. From the figures it is very clear that the performance of local searches varied even though all of them start with the same initial guess provided by RGA. The hybrid approach using CG (Fletcher-Reeves conugate gradient method) local search, performed better than other approaches. This is mainly because it uses the gradient information for finding the search direction. This underscores the need for having multiple local search methods as these methods are sometimes problem dependent. Note that we did not experiment with optimizing the location and number of observation points in these analyses. With careful placement of observation wells we could have reduced the number of observation locations and frequency of observatiobns. This is should be considered as a topic for future research. In practice, the observation wells should be placed judiciously to maximize detection and minimize cost.

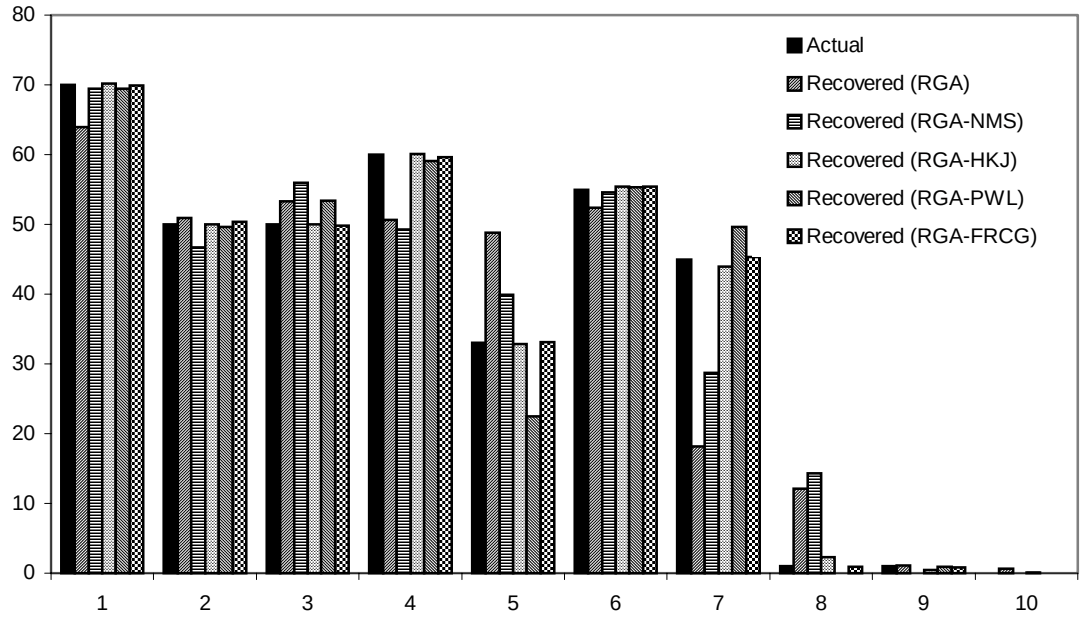



**Figure 5.15 (a) Performance of the hybrid approach for a single source release history reconstruction problem without noise.**

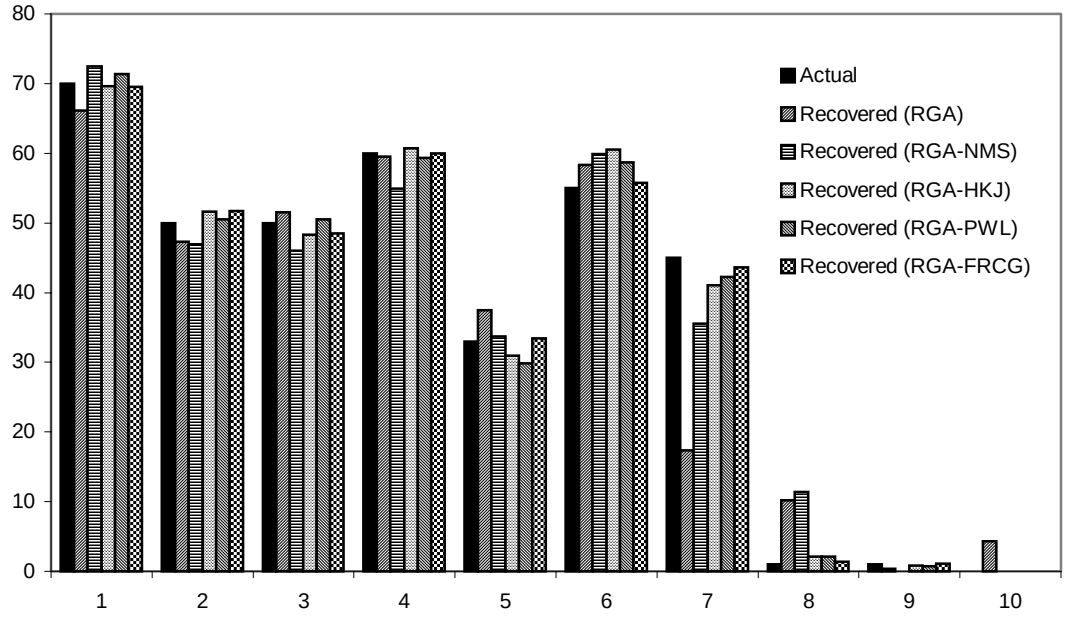



**Figure 5.15 (b) Performance of the hybrid approach for a single source release history reconstruction problem with 10% random white noise in observation data.**

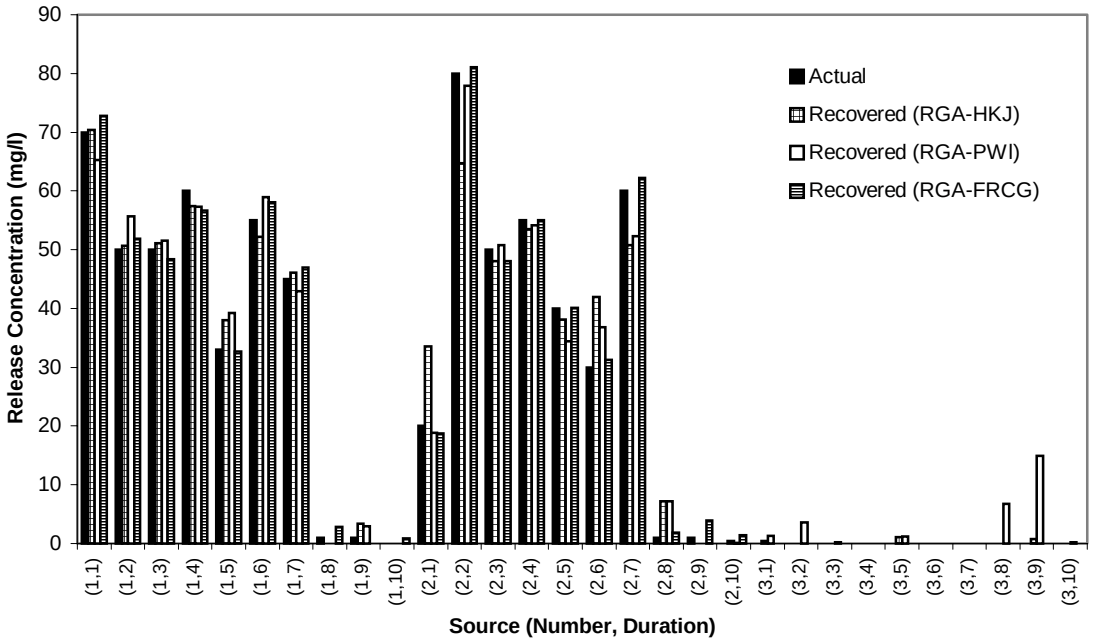



**Figure 5.16 (a) Performance of the hybrid approach for three sources release history reconstruction problem without noise in observation data.**

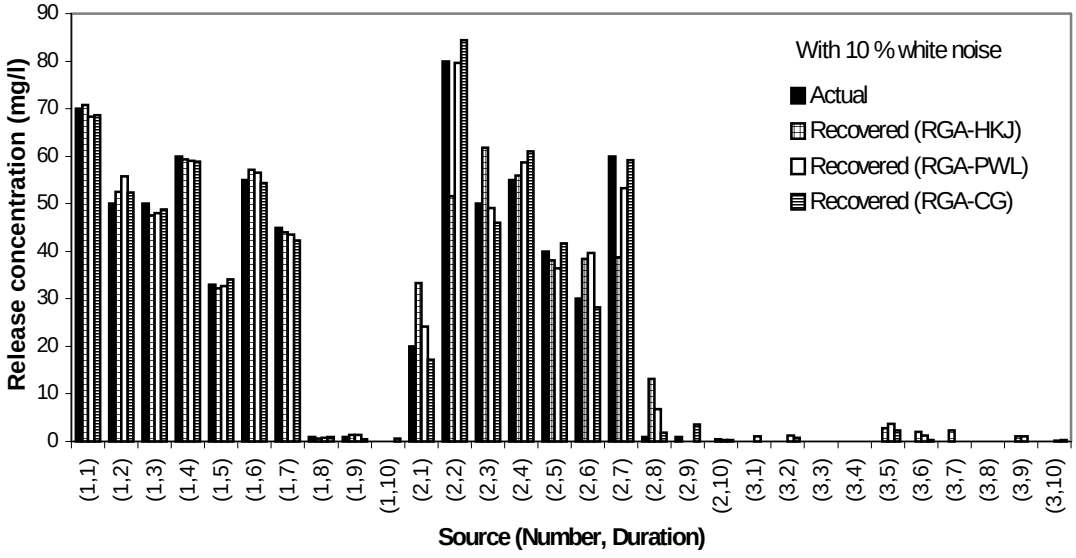



**Figure 5.16 (b) Performance of the hybrid approach for three sources release history reconstruction problem with 10% white noise in observation data.**

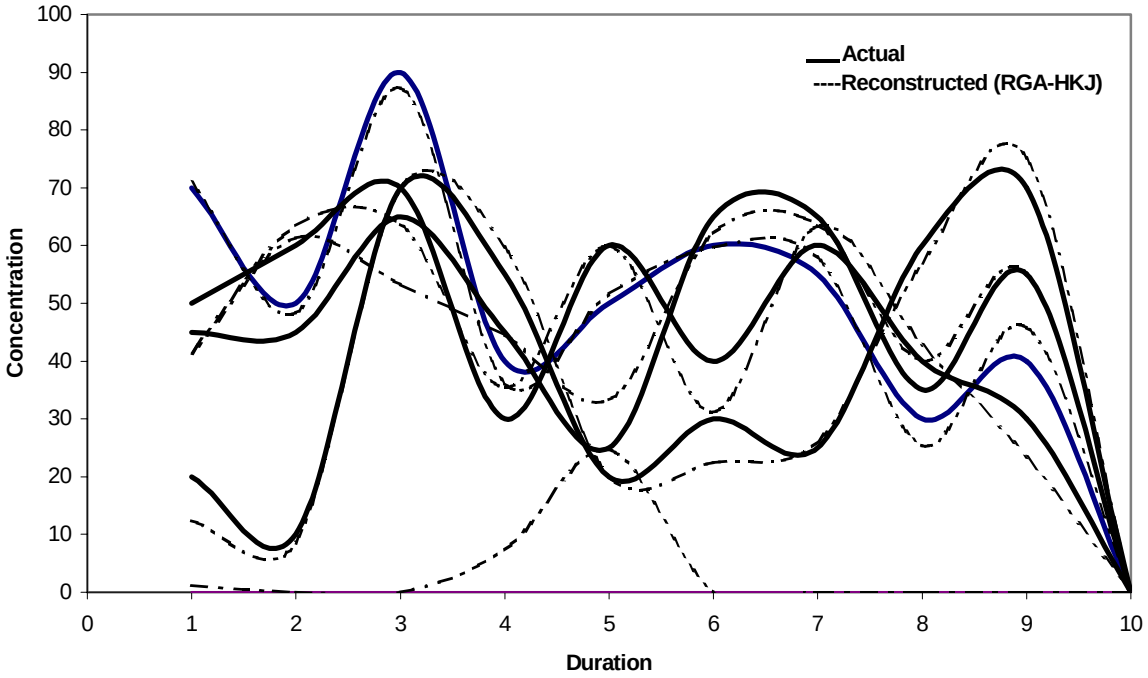



**Figure 5.17 (a) Performance of the hybrid (RGA-HKJ) approach for five-source release history reconstruction problem. Each curve represents a source.**

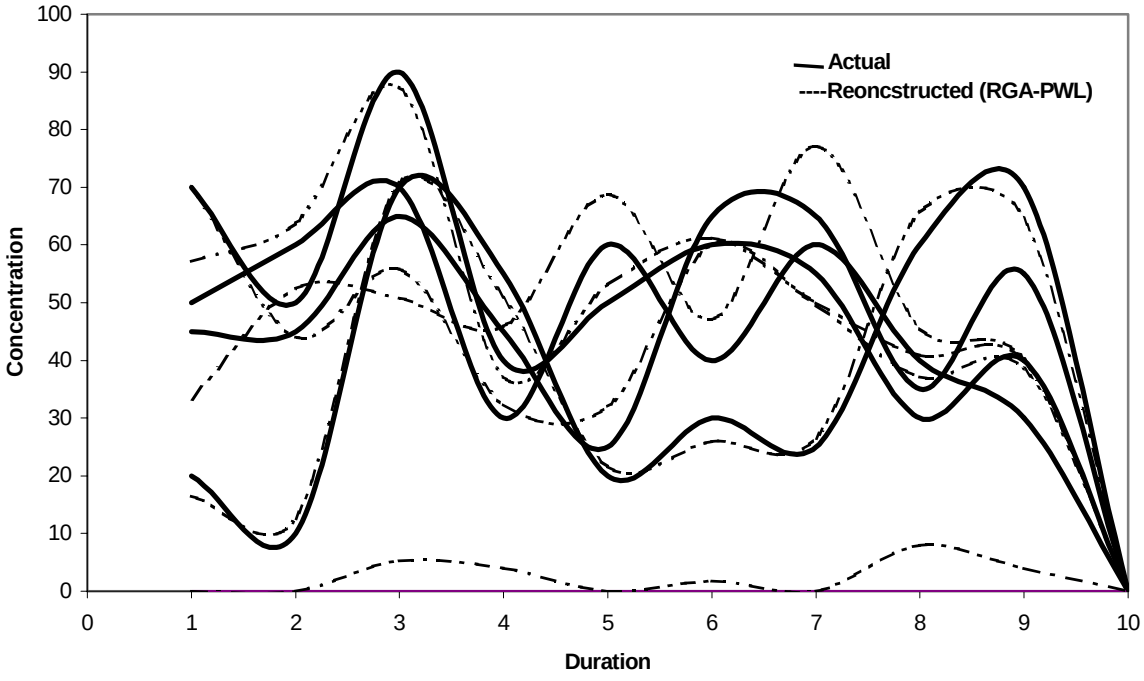



**Figure 5.17 (b) Performance of the hybrid (RGA-PWL) approach for five sources release history reconstruction problem. Each curve represents a source.**

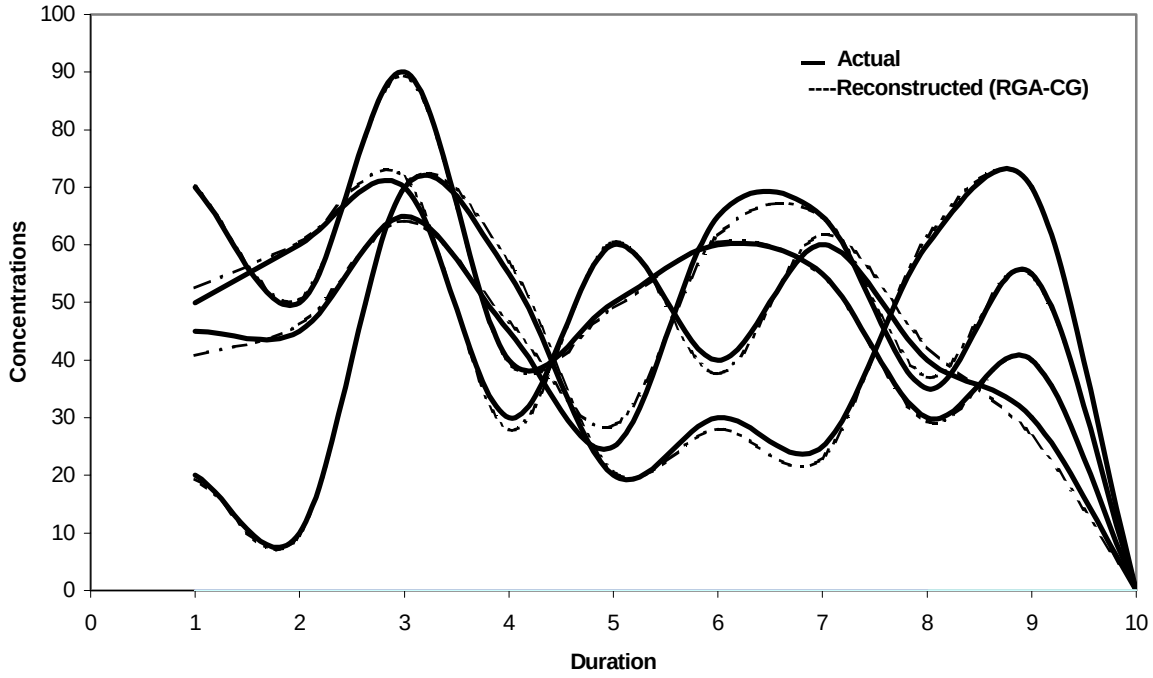



**Figure 5.17 (c) Performance of the hybrid (RGA-C.G) approach for five sources release history reconstruction problem. Each curve represents a source.**

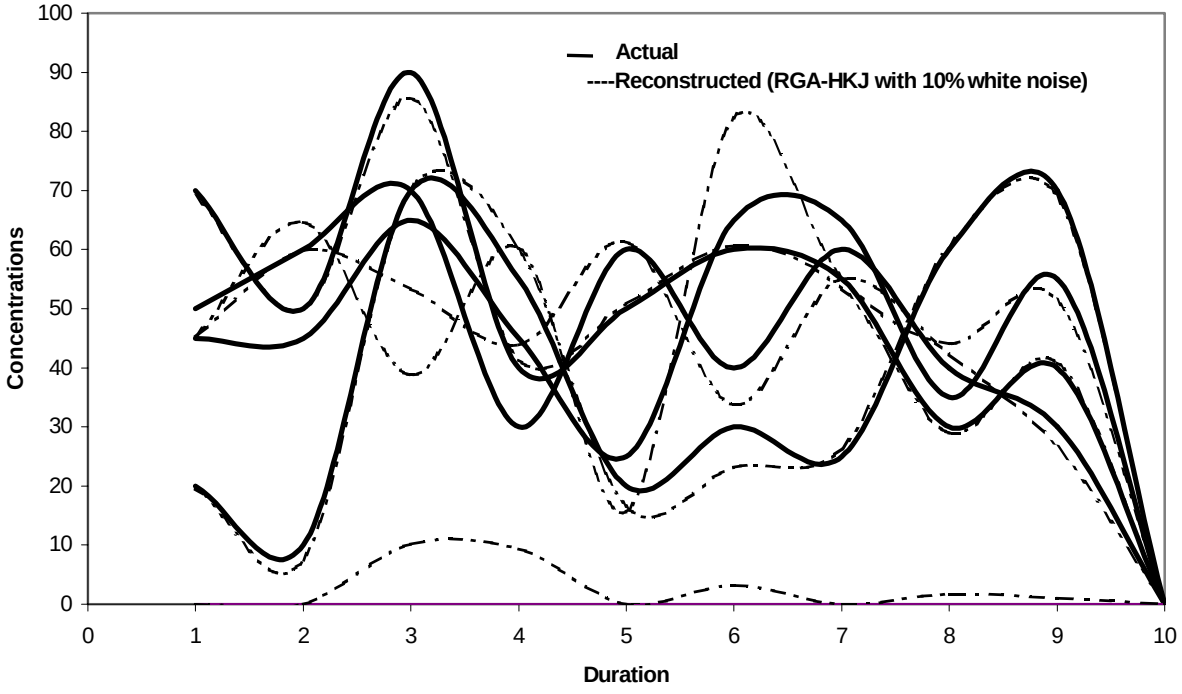



**Figure 5.18 (a) Performance of the hybrid (RGA-HKJ) approach for five sources release history reconstruction problem with 10% random white noise added to the observation data. Each curve represents a source.**

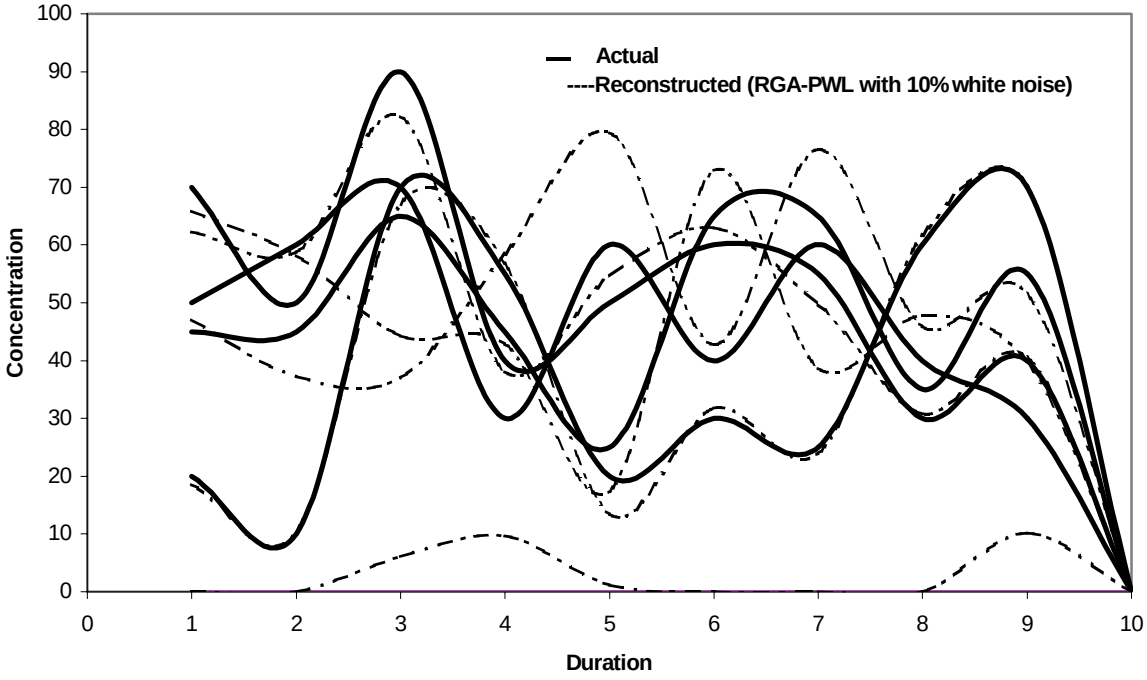



**Figure 5.18 (b) Performance of the hybrid (RGA-PWL) approach for five sources release history reconstruction problem with 10% random white noise added to the observation data. Each curve represents a source.**

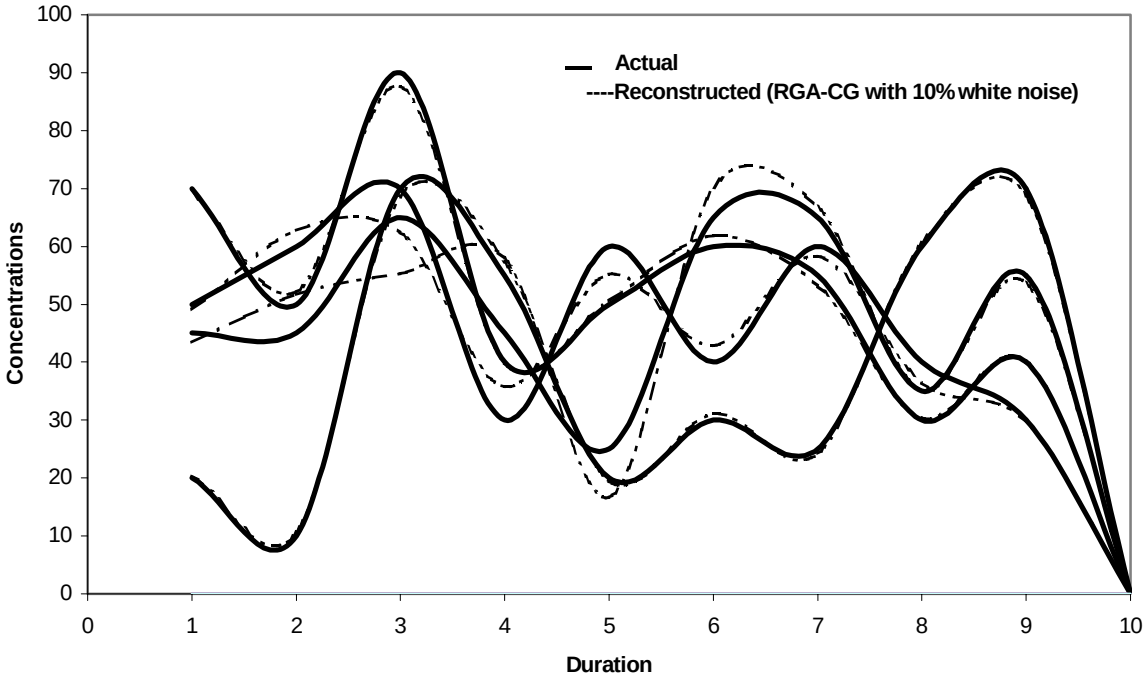



**Figure 5.18 (c) Performance of the hybrid (RGA-C.G) approach for five sources release history reconstruction problem with 10% random white noise added to the observation data. Each curve represents a source.**

# CHAPTER 6 - PARALLEL ARCHITECTURE AND PERFORMANCE

An emerging trend in the supercomputer market is to build parallel systems using commodity off the shelf (COTS) components. For example, SMP cluster architectures having more than thousands of processors can be formed by connecting hundreds of commodity SMP (symmetric multi processor) nodes, with each SMP node using a moderate number of processors (typically 2 to 32 processors per node). The processors used in these systems can be either vector processors (e.g. earth simulator[1]) or super scalar RISC (reduced instruction set computer) processors (e.g. IBM SP's, teragrid Linux clusters[2] etc). The programming model suited for many of these cluster systems is *message passing*, although shared memory programming may be used within an SMP node. Even though the shared memory programming paradigm is easy to program, it is severely limited by the number of processors available in the SMP node (or system). It should be noted that while message-passing model could also be used within the processors of an SMP node, it might not be the most covenient way of parallel programming. A hybrid approach using shared memory programming within a SMP node and message passing between nodes is also attractive.

As mentioned earlier our GA-LS-FEM implementation uses message-passing paradigm (MPI library) and has been ported to different architectures. Most of the simulations for this research are performed on the IBM SP3 with exception of the grid computing simulations, which are performed on the TeraGrid clusters at National Center for Supercomputing Applications (NCSA) and San Diego Supercomputing center (SDSC). A brief description of these machine architectures is given in the following sections.

[1] http://www.es.jamstec.go.jp/esc/eng/
[2] http://www.teragrid.org/

### 6.1 IBM SP3

This cache based supercomputer from IBM uses 64 bit POWER3-II processors running at 375 MHz with different processor counts (2, 4, 6, 8 or 16) per node. The theoretical peak performance of each processor is 1.5 Gflops. The processors super scalar architecture is capable of performing up to 8 instructions per clock cycle (2 floating-point, 2 single-cycle integer, 1 multi-cycle integer, 2 load/store, and 1 branch). The processor has 2 floating point units (FPUs), 3 fixed point execution units (FXU), two load/store units, and 3 levels of cache (Figure 6.1). The level-1 (L1) on chip cache is 64 KB and takes only 1 clock cycle to access. Each cache line is 128 bytes long. The L1 cache is very small (for data sets used in scientific applications) and to fix this an L2 cache is installed between L1 cache and main memory. The L2 cache is 8MB and is slower than L1 cache at 6 or 7 cycles. If there is a cache miss from both L1 and L2 caches, data has to be retrieved from main memory and the penalty for this is 36 cycles. The unified instruction and data cache (L2) sizes vary from 1 MB to 16 MB. The memory configuration based on user requirements can be any where from 256 MB to 16 GB or more. Each node has local disk space (~5 GB or more) and can be used as temporary scratch space. Users home directories are in the distributed file system on other servers. The disk access is much slower compared to memory access. Hence the users should try to store temporary data in memory to avoid frequent disk accesses, which may slow down the code. Figure 6.1 shows the functional units of this processor.

The performance of most parallel applications depends on the communication network condition. The SP node has two separate connections. The first is a standard 100 Mbps (mega-bits per second) Ethernet interface that connects each node to the outside world. The second interface is the SP Switch, which is a fast private network whose sole purpose is to carry data from parallel computations among the nodes. The peak bandwidth for the 100 Mbps Ethernet is around 90 Mbps and the lowest latency is 50 μs, although these numbers can vary greatly depending on the type and number of network switches traversed. The SP switch has a peak advertised performance of 1200 Mbps bandwidth and 1.2 μs latency, although only a peak performance of about 1000 Mbps

bandwidth and 26 μs latency has been reported using MPI. Thus the SP Switch has a significant performance advantage over Ethernet for parallel computations[3].

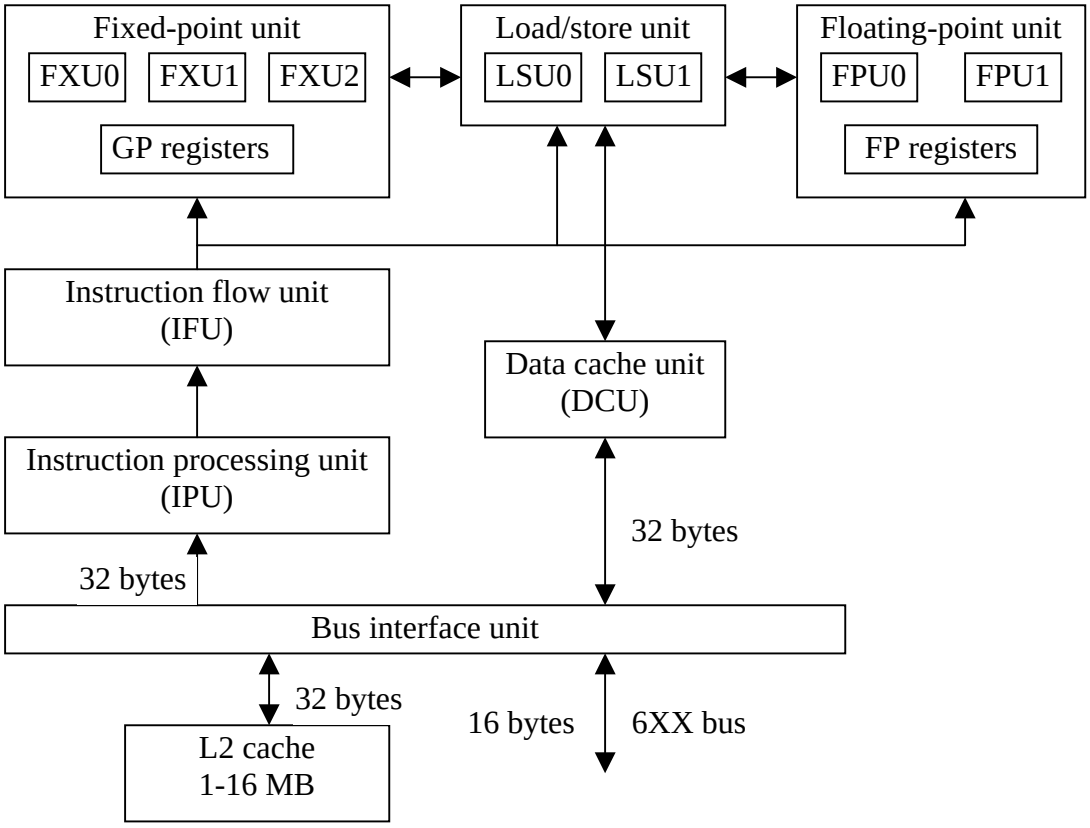



**Figure 6.1 Block diagram showing functional units of a power3-II architecture.**

### 6.2 Teragrid Itanium2 clusters

The teragrid Linux clusters at National Center for Supercomputing Applications at Illinois (TG-NCSA) and San Diego Supercomputer Center (TG-SDSC) are used in grid computing investigations (see section 6.4). These clusters are 256 or 128 dual processor systems using Intel Itanium2 1.3 GHz, 3MB integrated L3 cache Madison chip. The L1 and L2 caches are small compared to the IBM SP3 at 32 KB (instruction and data) and 256 KB respectively. The system bus is 400 MHz, 128 bits wide with a bandwidth of 6.4 GB/sec. The theoretical performance of this processor is 5.2 Gflops. The dual processors

[3] http://www-rcd.cc.purdue.edu/Performance/power3/hardware.html

nodes are connected by Myrinet™ interconnect and other networking to file system servers uses Gigabit Ethernet. The storage channel is reported as fiber channel. The peak theoretical bandwidth reported by vendors (Myricom[4]) is one way 2 Gbits/s and short messages latencies of 4 μs.

### 6.3 Parallel performance

Parallel performance of the implementation has been studied for the IBM SP3 architecture for speedup and load balance. The code has also been ported to SGI Origin 2400 and Compaq EV6 architectures.

#### 6.3.1 *Speedup*

A measure of parallel efficiency is speedup. Figure 6.2 shows the fine grained standalone FEM simulation performance in terms of speedup. The impact of fine-grained parallelism on the overall GA-FEM efficiency was performed by using different number of processors (e.g., 1, 2, 4 and 8 processors) for each FEM forward function evaluation (see figure 6.3). The best performance was observed for 2 processors per case, as cache benefits outweigh the communication costs. The fine-grained FEM parallelism is especially useful for inherently sequential LS methods (PWL and HKJ), as the function evaluation in LS process can be performed using more number of processors (say 8). The impact of coarse-grained parallelism is studied by measuring the total runtime for 10 GA generations with increasing number of processors on the IBM SP3 (see figure 6.4). Increasing the number of processors from 2 to 128 we see a speedup of 58 against an ideal linear speedup of 64. These simulations used a fixed fine-grained parallelism of 2 processors per FEM simulation. The coarse-grained parallelism in the GA optimizer (embarassingly parallel) is beneficial for using large number of processors and helps achieve very good parallel efficiency.

[4] http://www.myricom.com/myrinet/overview/

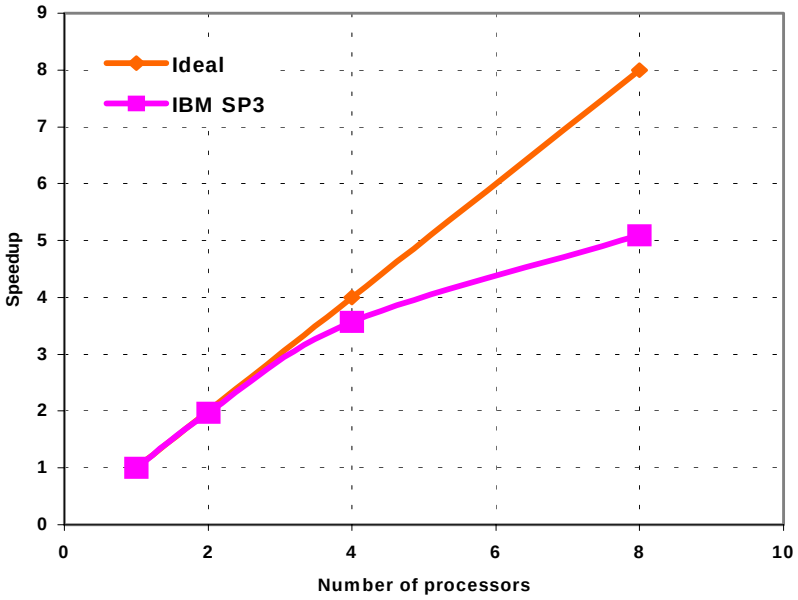



**Figure 6.2 Standalone fine grained FEM simulation performance.**

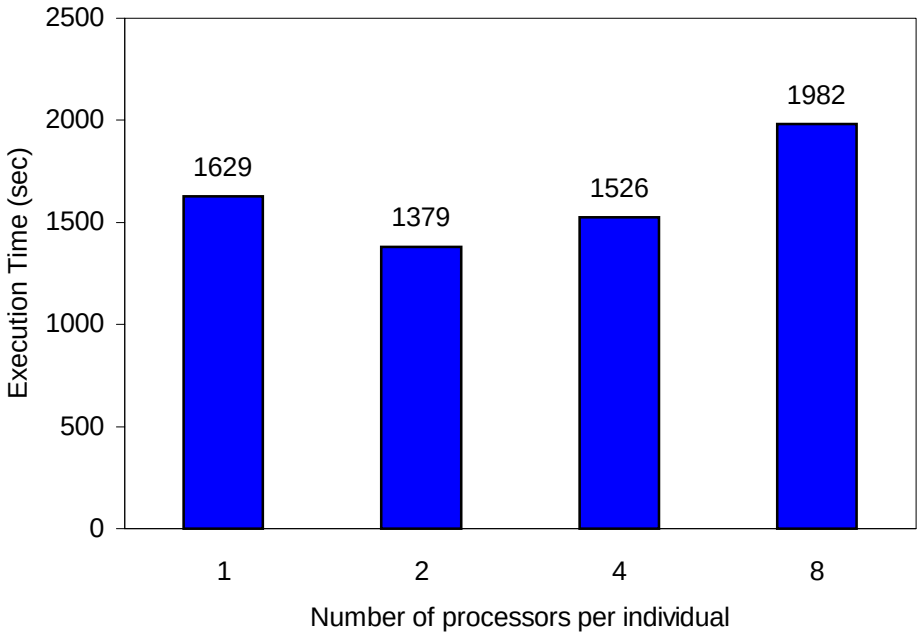



**Figure 6.3 Total time taken by 129 processors to complete 10 generations using 1, 2, 4, and 8 processors per individual on the IBM SP. Population size is 128.**

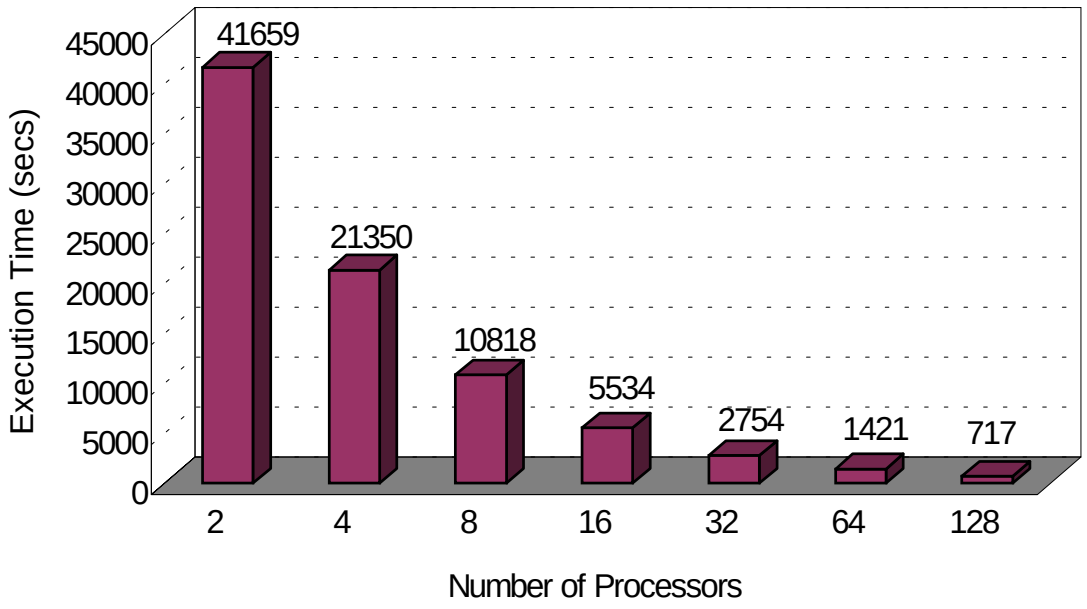



**Figure 6.4 Time taken for 10 GA generations on the IBM SP using 2 processor per FEM simulation**

### 6.3.2 *Load balance*

The load balance study is performed by artificially simulating the processor speed by forcing different processor groups to perform different number of FEM time steps. Four classes of processors are created and each class performs 50, 100, 150, and 200 time steps respectively. The processor class that performs 50 time steps will complete a transport simulation approximately 4 times faster than the processor class that performs 200 time steps. This way we simulate different processor speeds on the same machine. A fixed population size of 256 is used and simulations run for one GA generation. A total of 5, 9 and 17 processors are used with 1, 2, and 4 processors per subgroup for function evaluations respectively. One processor (master) does the GA computations. Figure 6.4 shows the load balance results. For perfect load balance each processor class should perform the same number of time steps. However, this is not generally feasible due to the startup round and the fact that each time step in the transport simulations may involve slightly different work load. Given these caveats, the load balance observed in Figure 6.4 is reasonably good illustrating the efficiency of the dynamic scheduling algorithm. For example, in the case with 2 processors per group, the processor class that performs 50

time steps complete 137 transport simulations (6850 time steps) and the group that performs 200 time steps complete 27 transport simulations (5800 time steps). Thus the processor class that is 4 times faster is performing about 5 times more work. The load balance efficiency (defined as the least number of time steps performed by a processor class divided by the most number of time steps performed by a processor class) is about 74%, 74%, and 69% for the 1, 2, and 4 processor groups respectively. While this is not perfect, it is still good given the caveats mentioned above.

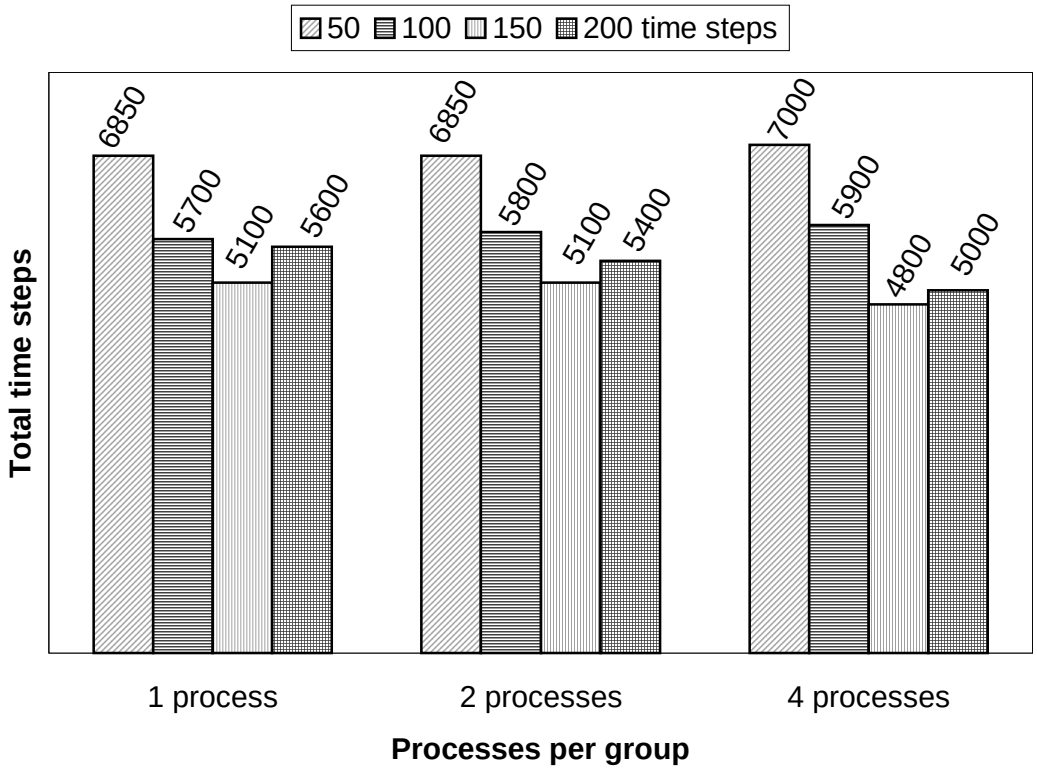



**Figure 6.5 Load balancing efficiency of self-scheduling algorithm. Processor speed is artificially simulated by varying number of time steps.**

### 6.3.3 *Scalability analysis of hybrid approach*

A simulation experiment to study the scalabilty of the hybrid approach by using the simplex method as the local search approach was conducted. As stated earlier, task parallelism in the simplex method is limited to the number of decision variables. The RGA with a population size of 100 is run for 50 generations and then the simplex method is continued for 100 cycles. The study was performed for 4 cases using total processor

counts of 17, 33, 65 and 129 for the RGA and simplex methods. The single source release history problem is tested (10 decision variables). The results are shown in figure 5.13. For RGA, 2 processors are used for each FEM simulation. For the simplex method, due to the reduction in coarse grained parallelism, the fine-grained FEM parallelism is increased to 4 or 8. The case using 129 processors takes 10000 seconds while the case using 17 processors takes 45000 seconds to complete the simulation (a speedup of 4.5 out of a possible maximum of 8). This shows that while the use of a local search inhibhits parallelim to some extent, by increasing the fine grained parallelism we are able to alleviate some of this.

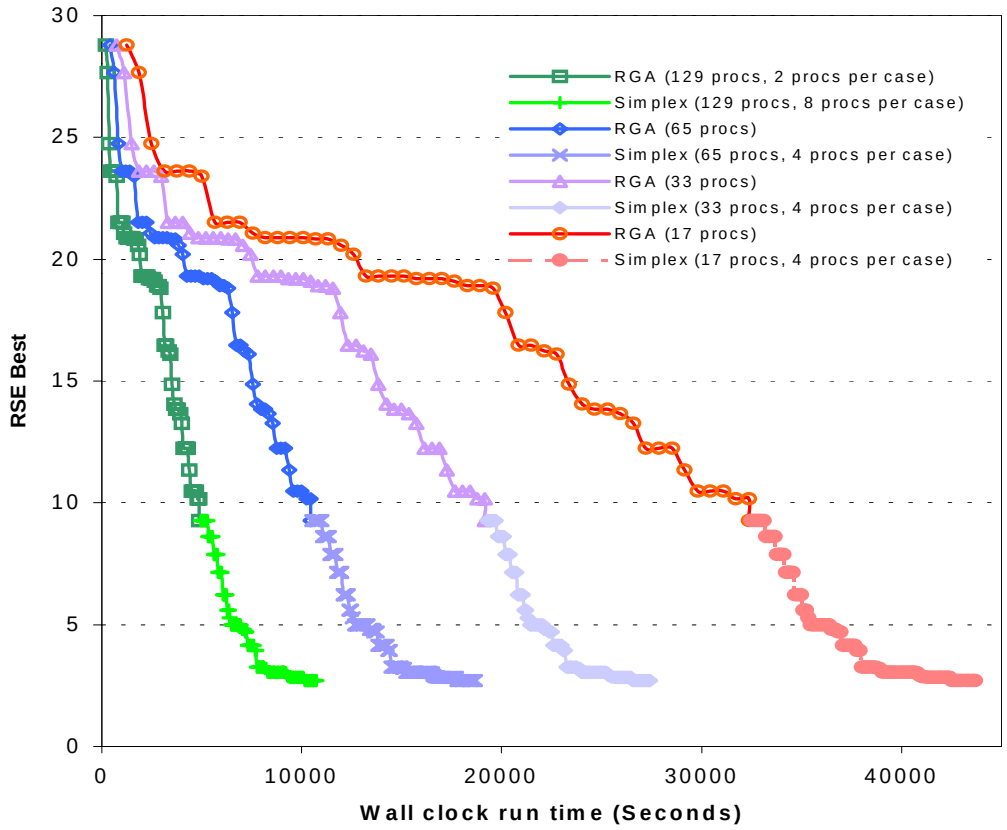



**Figure 6.6 The scalability of hybrid approach using different processor count for local search, namely simplex.**

## 6.4 Grid computing

The grid computing investigations are carried on the *TeraGrid*. The National Science Foundation (NSF) funded TeraGrid project is a collection of heterogeneous geographically distributed high performance computer resources (typically parallel supercomputers) connected by high bandwidth and low latency network. The coarse-grained task parallelism and fine-grained data parallelism make this application ideal for the grid environment since the application has two-levels of communication hierarchy (one loosely coupled and the other tightly coupled) analogous to the hardware characteristics of the grid. TeraGrid is continually evolving and currently five sites, National Center for Supercomputing Applications (NCSA), San Diego super computer center (SDSC), Pittsburgh Supercomputing Center (PSC), Argonne National labs (ANL), and California Institute of Technology (Caltech) are part of the TeraGrid. Several new sites are to be added in the near future. At the time of this writing the systems are still in testing phase and are not available to general users for production runs. The accounts on these machines were available for testing only in August 2003. Therefore the systems are not fully stable and the tests are carried only at NCSA and SDSC sites. The TeraGridLinux clusters at NCSA (TG-NCSA) and SDSC (TG-SDSC) are similar systems with 256 and 128 dual processor nodes respectively. For hardware information see section 6.2. Additional hardware and software features of the TG machines are available on the internet[5].

The first step was to port the GA-LS-FEM code to the NCSA and SDSC TeraGrid Linux clusters. It primarily required minor compilation error fixes. Extensive single site runs and limited number of cross-sites were done to evaluate performance. Recall that our GA-LS-FEM implementation has both fine and coarse-grained levels of parallelism. The fine-grained parallelism (FEM transport simulator) computations are communication intensive. Thus for cross-site runs, the processors forming the FEM subgroups are ensured to be local to a single site and are not across sites (section 4.2).

[5] www.teragrid.org

### 6.4.1 *Results*

During this investigation MPICH-G2 (Globus flavor of MPICH) was unavailable (see section 4.2). Thus Virtual Message Interface (VMI) flavor of MPICH (MPICH-VMI-INTEL) is used in the runs. The two sub sections that follow give the bandwidth benchmark test results and the FEM/RGA-FEM simulation results.

***Teragrid bandwidth (ping-pong) test***

Ping-pong MPI bandwidth tests are performed for single cluster (both intranode and internode) and cross-cluster cases. The interconnect/network bandwidth is measured by sending and receiving messages of varying sizes between two processors using MPI_send/MPI_Recv. This test provides information about the actual available bandwidth if one were to run a real application. Table 6.1 gives the single site intra and inter node interconnect bandwidths and the cross-site network bandwidth. While the MPI bandwidth increases with message size, a peak intranode bandwidth of 1.5 GB/sec (for 256 KB messages) and a peak internode bandwidth of 0.23 GB/sec (for 512 KB messages) is observed. The cross-site network bandwidth peaks at 42 MB/sec for 4MB size messages. This is very small compared to theoretical network bandwidth of 40Gbits/sec. The network latency between SDSC and NCSA has been measured to be around 30ms and is very high. Hence, for cross site runs proper care must be taken to reduce the number of small messages sent across and bundle the messages if necessary. Also latency hiding mechanisms such as overlapping computations with communication should be considered to further improve performance.

**Table 6.1 MPI Bandwidth for single and cross site run**

| Bytes | Single Site | | Cross Site (MB/sec) |
|---|---|---|---|
| | Intranode (MB/sec) | Internode (MB/sec) | |
| 1024 | 245.92 | 50.82 | 0.03 |
| 4096 | 596.83 | 103.02 | 0.093 |
| 8192 | 788.5 | 128.78 | 0.1819 |
| 16384 | 720.51 | 146.85 | 0.2738 |
| 32768 | 1016.64 | 180.39 | 0.4274 |
| 65536 | 1262.05 | 203.44 | 0.8622 |
| 131072 | 1429.44 | 217.22 | 2.2856 |
| 262144 | 1513.78 | 224.87 | 3.4516 |
| 524288 | 1492.93 | 228.84 | 7.3102 |
| 1048576 | 942.22 | 231.11 | 11.8127 |
| 2097152 | 659.67 | 233.39 | 29.7817 |
| 4194304 | 649.45 | 234.43 | 42.7302 |

***FEM and RGA-FEM performance on TeraGrid***

The performance of standalone FEM code is compared in figure 6.7 for single-site and cross-site runs. While we see a superlinear speedup behavior for runs within a cluster, the cross-site FEM runs show a very large communication overhead. This is mainly due to the high network latency (30ms) between TG-NCSA and TG-SDSC. For example, for the 8 processors per FEM simulation case single cluster run takes only 8 seconds whereas the cross-site run takes approximately 1300 seconds. This clearly highlights the importance of performing FEM simulations locally on a single cluster.

RGA-FEM performance results are given in table 6.2. The runs using MPICH-GM-INTEL perform better in most cases than the MPICH-VMI-INTEL for single and cross-site runs since the GM interface uses the faster Myrinet connection within cluster instead of TCP throughout.

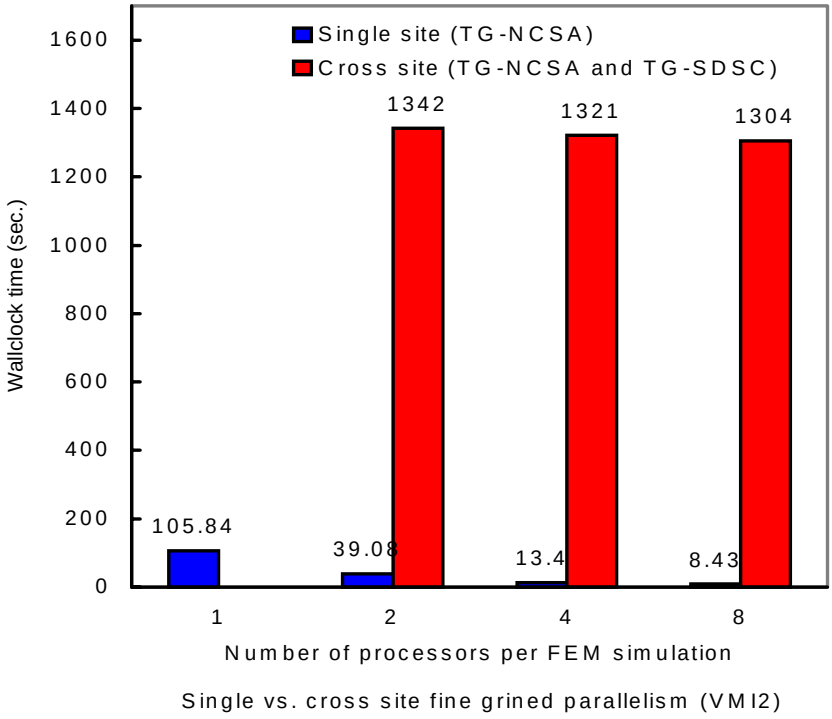



**Figure 6.7 Performance of single vs. cross-site fine grained parallelism**

**Table 6.2 Runtimes for single and cross-site RGA-FEM simulations (using VMI1)**

| **Number of processors for RGA-FEM simulation** | **Number of processors per FEM simulation** | | | |
|---|---|---|---|---|
| | 1 | 2 | 4 | 8 |
| | **Single site runtime (sec) using VMI1+TCP(Gig) interconnection** | | | |
| 17 | 848 | 681 | 966 | - |
| 33 | 474 | 366 | 438 | 870 |
| 65 | 259 | - | 230 | 467 |
| 129 | | 122 | 136 | 268 |
| | **Single site runtime (sec) using VMI1+GM interconnection** | | | |
| 17 | 834 | 596 | 351 | 491 |
| 33 | 475 | 316 | 201 | 276 |
| 65 | 268 | 174 | 125 | 168 |
| 129 | | 104 | 53 | 67 |
| | **Cross site runtime (sec) using VMI1+TCP(Gig) interconnection** | | | |
| 17 (8 + 9) | 835 | 700 | 1215 | 1937 |
| 33 (16 + 17) | 478 | 369 | 421 | 890 |
| 65 (32 + 33) | 261 | 217 | 390 | 667 |
| 129 (64 + 65) | | 131 | 183 | 428 |

# Chapter 7 – Research contributions and topics for further research

This research has centered on three areas of exploration: (i) development of *optimization* algorithms for inverse modeling, (ii) *application* to various groundwater inverse problems, and (iii) enabling *parallel computing* technologies for the compute intensive simulations. Emphasis is given in the following order: optimization, parallel computing, and application. This chapter provides a summary of findings, accomplishments, and areas for future research.

## 7.1 Research findings

- Hybrid optimization approaches are generally more effective than standalone GA or LS for inverse modeling.
- Release history problems are easier to solve since the signature of the decision variables are more effectively carried in the break through curves. For source identification problems, the location information is less effectively carried by the signals and are more difficult to solve.
- The biological activity zone identification problems are more naturally posed as discrete optimization problems and are less suited for real GA's or local searches.
- The two level of parallelism encountered in these types of problems, namely, coarse grained in the optimizer and fine grained in function evaluation is ideal for grid environments.
- The middleware tools (or technology) for grid computing is not yet mature and needs further development for more wide spread acceptance.

## 7.2 Research accomplishments

- An efficient and flexible parallel hybrid optimization framework has been developed for solving inverse problems [Sayeed and Mahinthakumar, 2002].
- The parallel implementation can handle multi-type/multi-start GA or local searches (LS) concurrently [Sayeed and Mahinthakumar 2003 (1) and Mahinthakumar and Sayeed, 2003].

- The following methods have been implemented in the optimization module: (i) Three variations of GA: binary GA, integer and real GA's, (ii) Three non-gradient based local search techniques: Nelder-Meade simplex (NMS), Hooke-Jeeves pattern search (HKJ), and Powell's method of conjugate directions (PWL), and (iii) One gradient-based method: Fletcher-Reeves conjugate gradient (CG) [Sayeed and Mahinthakumar, 2003 (3)].
- The optimization module has been optionally configured to perform noisy-GA and modeling to generate alternatives (MGA) simulations.
- The implementation is portable to most parallel environments as it is written in Fortran 77/90 and parallelized using the message passing interface (MPI) library.
- The GA-LS-FEM framework has been tested on the Teragrid [Sayeed and Mahinthakumar 2003 (2)].
- The parallel optimization framework has facilitated solutions to complex three-dimensional groundwater inverse problems that have not been attempted before.

### 7.3 Topics for further research

- Additional optimization algorithms can be added to the existing framework.
- Preliminary investigations of Noisy-GA approach to handle uncertainty in groundwater problems needs further study. Also, it is necessary to identify the best possible approach to handle uncertainty during local searches.
- The optimization and function evaluation (groundwater FEM simulator) codes are tightly coupled in this implementation for improved portability at the expense of modularity. However, future implementations should look into more elegant formulations that can use two separate executables that communicate to one another via the new features of MPI-2.
- The modeling to generate alternatives (MGA) approach to address non-uniqueness nature of some of the groundwater inverse problems should be explored further with new test problems.
- The optimization approaches should be studied for other problems in the groundwater area such as hydraulic conductivity inversion.

- The hybrid optimization approaches developed in this thesis should be tested for real field characterization problems.

# BIBLIOGRAPHY


Abramson D, Rajkumar Buyya, Jonathan Giddy, 2002. **A computational economy for grid computing and its implementation in the Nimrod-G resource broker**, **,** Future generation Computer System 18 (2002) 1061-1074.

Aral, M.M., and J. Guan, (1996). **Genetic algorithm in search of groundwater pollution sources**, in Advances in Groundwater Pollution Control and Remediation, edited by M.M. Aral, NATO ASI Series, 2(9), 347-369.

Aral, M.M., J. Guan and M L Maslia, (2001). **Identification of contaminant source location and release history in aquifers**, Journal of hydrologic engineering, Vol. 6, No. 3, May/June, 2001.

Atmadja J and Bagtzoglou AC, (2001). **State of the art report on mathematical methods for groundwater pollution source identification**, Environmental Forensics, 2 (3): 205-214 SEP 2001.

Belegundu D. Ashok and Chandrupatla R. Tirupathi, 1999. **Optimization concepts and applications in engineering, Prentice Hall Inc.**

Brill E.D. Jr., J.M. Flach, L.D. Hopkins, and S. Ranjithan, (1990). **MGA: A Decision Support System for Complex, Incompletely Defined Problems,** IEEE Trans. on Systems, Man, and Cybernetics, vol. 20, no. 4, pp. 745-757, 1990.

Cieniawski S. E., Eheart, J.W., Ranjithan, S. (1995). **Using genetic algorithms to solve a multi-objective groundwater monitoring problem**, Water Resources Res., 31(2), 399-409.

Davis, L. (ed). (1991). **Handbook of genetic algorithms**. Van Nostrand Reinhold, New York, N.Y

Espionza, P. F, B. S. Minsker and D. E. Goldberg (2003). **An adaptive hybrid genetic algorithm for groundwater remediation design**, Journal of water resurces planning and management ASCE (accepted 2003).

Floudas, C.A., and Pardalos, P.M., (2001). **Encyclopedia of Optimization**, v.1-6, Kluwer Academic Publishers, Dordrecht and London.

Giest, A., et al. (1994). **PVM: Parallel Virtual Machine – A uers guide and tutorial for network computing**. MIT Press, Cambridge, Mass.

Giacobbo F, Marseguerra M, and Zio E, (2002). **Solving the inverse problem of parameter estimation by genetic algorithms: the case of a groundwater contaminant transport model**, Annals of Nuclear Energy, 29 (8): 967-981 MAY 2002.

Goldberg, D., (1989). **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison-Wesley, 412 pp.

Goldberg, D.E., and K. Sastry (2001). **A practical schema theorem for genetic algorithm design and tuning**, IlliGAL Report No. 2001022, University of Illinois at Urbana-Champaign.

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

Gorelick, S. M. **A review of distributed parameter groundwater management modeling methods**. Water Resources Research, 19(2), 305-319, 1983.

Gropp, W., Lusk W., and Skjellum A., (1999). **Using MPI: Portable Parallel Programming with the Message-Passing Interface**, 2nd edition, The MIT Press, Cambridge, MA.

Heidari, M and S. Ranjithan, (1998). **A hybrid optimization approach to the estimation of distributed parameters in two-dimensional confined aquifers**, Journal of American Water Resources Association, 34(4), pp. 909-920, 1998.

Hilton, Amy B. Chan and T. B. Culver (2003). **Groundwater remediation design under uncertainity using genetic algorithms**, Journal of water resurces planning and management, ASCE (accepted 2003).

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

Huang CL, and Mayer AS, (1997). **Pump-and-treat optimization using well locations and pumping rates as decision variables**, Water Resources Research, 33 (5): 1001-1012 MAY 1997.

Johnston E Williams (2002). **Computational and data Grids in large scale science and engineering,** Future Generation Computer Systems 18 (2002) 1085-1100.

Katsifarakis, K. L, Karpouzos, D.K and N. Theodossiou, (1999). **Combined use of BEM and genetic algorithm in groundwater flow and mass transport problems**, Engineering Analysis with Boundary Elements, 23 (1999) 555-565.

Kennedy, J., and Eberhart, R.C. (1995). **Particle swarm optimization.** In proceedings of IEEE International Conference on Neural Networks, Perth, Australia, IEEE Service Center, Piscataway, NJ., pp. 1942-1948.

Kohl, J. A., (1997) **High Perfomance Computing: Innovative Assistant to Science.** ORNL Review Volume 30, Numbers 3 & 4, 1997

Liu C. and W. P. Ball, (1999). **Application of inverse methods to contaminant source identification from aquitard diffusion profiles at Dover AFB, Delaware**, Water Resources Research, 35(7), 1975-1985, 1999.

Loughlin, D. H., and S. Ranjithan, (1999). **Chance-constrained genetic algorithms**, Genetic and Evolutionary Computation Conference (GECCO), p.369-376.

Loughlin, D. H., S. Ranjithan, J. W. Baugh, and E. D. Brill, (2001). **Genetic Algorithm Approaches for Addressing Unmodeled Objectives**, Engineering Optimization, 33 (5): 549 – 569, 2001.

Natarajan Anand, Marty Humprey, Andrew S. Grimshaw, (2002). **The Legion support for advanced parameter-space studies on a grid,** Future generation Computer System 18 (2002) 1033-1052.

Mahar PS, and Datta B, (1997). **Optimal monitoring network and ground-water-pollution source identification**, Journal Of Water Resources Planning And Management-ASCE, 123 (4): 199-207 JUL-AUG 1997.

Mahar PS, and Datta B, (2000). **Identification of pollution sources in transient groundwater systems**, Water Resources Management, 14 (3): 209-227 JUN 2000.

Mahar, P.S., and B. Datta, (2001). **Optimal identification of groundwater pollution sources and parameter estimation**, ASCE Journal of Water Resources Planning and Management, 27(1), 20-29.

Mahinthakumar G., and Saied F. (1999). **Implementation and Performance Analysis of a Parallel Multicomponent Groundwater Transport Code**, CD-ROM Proceedings of the 1999 SIAM Parallel Processing Meeting, San Antonio, TX, March 1999.

Mahinthakumar G., Gwo J.P., Moline G.R., Webb O. F., (1999). **Subsurface biological activity zone detection using genetic search algorithms**, ASCE Journal of Environmental Engineering. Vol. 125, No. 12, p.1103-1112. December 1999.

Mahinthakumar, G., (1999). **PGREM3D: Massively Parallel Codes for Groundwater Flow and Transport**, Unpublished Report.
http://www4.ncsu.edu/~gmkumar/pgrem3d.pdf .

Mahinthakumar,G., and Sayeed, M. (2003). **Hybrid genetic algorithm - local search approaches for groundwater source identification problems,** Special Issue on Evolutionary Computation, ASCE J. of Water Resources Planning and Management, 2003.

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

McLaughlin D, and Townley LR, (1996). **A reassessment of the groundwater inverse problem**, Water Resources Research, 32 (5): 1131-1161 MAY 1996.

Michalewicz, Z. (1996). **Genetic algorithms + data structures = evolution programs**, 3rd Edition, Springer-Verlag, New York, 387 p.

Miller, B. L and D. E. Goldberg, (1996), **Optimal Sampling for Genetic Algorithms**, Intelligent Engineering Systems through artificial Neural Networks (ANNIE' 96), Vol. 6, pp 291-298, NewYork, ASME Press, 1996.

Miller, B.L. (1997). **Noise, sampling and genetic algorithms**, PhD Thesis, University of Illinois at Urbana-Champaign.

Morrison, R.D. (2000). **Application of Forensic Techniques for Age Dating and Source identification in environmental litigation**, Environmental Forensics, 1, 131-153.

National Research Council, (1990). **Groundwater models – scientific and regulatory applications**, National Academic Press, Washington, D.C.

Neupauer, R. M., B. Borchers, and J. L. Wilson, (2000). **Comparison of inverse methods for reconstructing the release history of a groundwater contaminant source**, Water Resources Research, 36(9), 2469-2475.

Pan LH, and Wu LS (1998). **A hybrid global optimization method for inverse estimation of hydraulic parameters: Annealing-simplex method**, Water Resources Research, 34 (9): 2261-2269 SEP 1998.

Powell, M. J. D, (1964). **An efficient method for finding the minimum of a function of several variables without calculating derivatives**, Computer Journal, Vol. 7, No. 4, p. 303-307, 1964.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. **Numerical recipes in Fortran, Second edition**. Cambridge University Press, New York, NY, 1996.

Quinn, M. J, **Parallel Programming in C with MPI and OpenMP**. –1st ed. 2004.

McGraw Hill, ISBN 0-07-282256-2.

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

Santamarina, J. C and Fratta, D (1998). **Introduction to Discrete Signals and Inverse Problems in Civil Engineering**, ASCE PRESS

Sayeed, M. and G. Mahinthakumar, (2002). **A multilevel parallelization scheme based on MPI communicators for solving groundwater inverse problems,** High performance computing 2002, Society for Computer Simulation International, p. 137-144

Sayeed, M. and Mahinthakumar .G (2003). **Hybrid optimization approaches for solving groundwater inverse problems in a parallel computing environment,** Accepted by World Water & Environmental Resources Congress 2003 & related symposia, ASCE EWRI conference.

Sayeed. M., Mahinthakumar .G, Dongju Choi, Leesa Brieger, Dominic Holland, Nick. Karonis, (2003). **Parallel hybrid optimization approaches for the solution of large**

**scale groundwater inverse problems on the teragrid,** Poster presentation at SC 2003, Phoenix AZ.

Sayeed, M. and Mahinthakumar .G (2003). **An efficient parallel optimization framework for solving inverse problems.** Draft in preparation for submission to ASCE Journal of Computing in Civil Engineering.

Sciortino, A., T. C. Harmon, and W-G Yeh, (2000). **Inverse modeling for locating dense nonaqueous pools in groundwater under steady flow conditions**, Water Resources Research, 36(7), 1723-1735, 2000.

Semprini L., and P.L. McCarty, 1991. **Comparison between model simulations and field results for in-situ biorestoration of chlorinated aliphatics: Part 1**. Biostimulation of methanotropic bacteria. Ground Water, 29(3), 365-374.

Shieh, Horng-Jer and Peralta, R. C, 2003. **Optimal in-situ bioremediation system design using parallel recombinative simulated annealing**, Journal of water resurces planning and management ASCE (in press 2003).

Skaggs, T.H., and Z. J. Kabala (1994), **Recovering the release history of a groundwater contaminant**, Water Resources Research, 3-, 71-79, 1994.

Smalley JB, Minsker BS, Goldberg DE, (2000). **Risk-based in situ bioremediation design using a noisy genetic algorithm**, Water Resources Research, 36 (10): 3043-3052 OCT 2000.

Sun, Ne-Zheng (1994). **Inverse problems in groundwater modeling**, **Theory and Applications of Transport in Porous Media** (Ed. Jacob Bear), Vol. 6, Kluwer Academic Publishers, 337 p.

Takewaki, I (2000) **Dynamic Structural Design, Inverse Problem Approach**, WIT PRESS.

Tompson, A.F.B., R. Aboubu, and L.W. Gelhar. (1989). **Implementation of the three-dimensional turning bands random field generator**. Water Resources Research. vol. 25, no. 10 (Oct.): 2227-2243.

Waldrop M. M, (2002) **Grid Computing could put the planet's information-processing power on tap**, Technology Review May 2002.

Wang M, and Zheng C, (1997). **Optimal remediation policy selection under general conditions**, Ground Water, 35 (5): 757-764 SEP-OCT 1997.

Wang M, and Zheng C, (1998). **Application of genetic algorithms and simulated annealing in groundwater management: formulation and comparison**. Journal of American Water Resources Association, 34(3), 519-530, 1998.

Woodbury A, Sudicky E, Ulrych TJ, Ludwig R (1998). **Three-dimensional plume source reconstruction using minimum relative entropy inversion**, Journal Of Contaminant Hydrology, 32 (1-2): 131-158 JUL 1998.

Woodbury AD, and Ulrych TJ, (2000). **A full-Bayesian approach to the groundwater inverse problem for steady state flow**, Water Resources Research, 36 (8): 2081-2093 AUG 2000.

Yoon JH, and Shoemaker CA, (2001). **Improved real-coded GA for groundwater bioremediation**, Journal Of Computing In Civil Engineering, 15 (3): 224-231 JUL 2001.

# Appendix - A

# PRELIMINARY INVESTIGATION OF NOISY-GA

The Noisy-GA approach uses multiple realizations (sampling) to reduce the amount of noise from fitness evaluations in noisy environments. This is similar to Monte-Carlo type sampling and provides a more realistic estimate of the fitness [Gopalakrishnan and Minsker 2001]. However, Monte-Carlo simulation modeling uses a large sampling set compared to a few samples per design used by noisy GA to identify robust designs. The noise in the system can arise from any factor that affects the accurate evaluation of fitness. These factors can be approximate fitness function, the use of noisy data, uncertainty in the input parameters and human error. For example, in groundwater modeling noisy fitness functions can arise from uncertainty in input parameters (e.g. hydraulic conductivity, porosity and chemical reaction rate coefficients) or measurement errors (e.g. hydraulic heads, flow rates, contaminant arrival times, solute concentrations, and/or mass removal rates at monitoring points).

## A.1 Current approach

Uncertainties in results can be associated with model input parameters or with numerical and conceptual difficulties present in the model [Zheng and Bennett, 2002]. While there are several factors that contribute to uncertainty, this study focuses on uncertainty resulting from the input hydraulic conductivity data. Several studies have identified this to be the most challenging parameter to predict correctly for any particular site [Zheng and Bennett, 2002].

Multiple realizations are used for the noisy-RGA approach. Random heterogeneous hydraulic conductivity fields (K-field) are generated using the 3D turning bands code. The turning bands code is a parallelized version of the original code developed by Andy Thompson (Thompson et al. 1989). Then the steady state flow fields (velocity and flux) are generated using the PGREM3D flow solver using the K-fields

generated. Since the sample size plays a critical role in obtaining a robust solution, several sample sizes are experimented.

For the noisy-RGA a sample set of 100 is chosen for generating the K-field. The 100 hydraulic conductivity fields are generated synthetically by running 100 Monte-Carlo simulations of the turning bands code. These 100 K-fields are used by the flow code to generate the 100 steady state flux and velocity fields. Several test cases, each with different sets of realizations are experimented to analyze the behavior of noisy-RGA's for a single source release history reconstruction problem. One of the K-field is chosen for generating the reference concentration profiles (observed values).

### A.2 Experiments

The experiments carried out are

- One randomly selected K-field for each individual in a population for every generation.
- 1, 2, 4, 8 and 16 K-field realizations for each GA generation (all individuals use the same set of realizations). The fitness for each individual is the average of the realizations.

While it is not clear how exactly one would proceed with the LS when using noisy-GAs, one could envision the following options:

- Using the same velocity and flux field as used for generating reference observed concentration values.
- Randomly selecting one of the velocity and flux fields from the sample set for the whole LS run.
- Randomly selecting any one of the velocity and flux fields from the sample set.
- Using the average of the sample set.
- Perform LS for each realization and then take the average as the solution.

### A.3 Results

From the results shown in table A.1 the solutions produced using multiple realizations are very different in the decision space but are close in their objective

function value. After the noisy-RGA approach, hybrid optimization using different local searches can be performed using the actual hydraulic conductivity field used in generating the observed concentration profiles or some other approach such as using the same number of realizations from the sample set for each forward function evaluation.

The post-processing of solutions of different realizations obtained by the noisy-RGA are necessary to estimate the degree of robustness of the solutions. This can be estimated by calculating the average RSE produced by the actual and noisy-RGA solutions (1, 2, 4, 8 and 16 multiple realization cases) against the full sample set (i.e. running with all the 100 hydraulic conductivity fields). For identifying a robust solution, one hypothesis is that the solution giving the least average RSE will be the robust solution. This would show the robustness of the solution against any perturbation or noise in the hydraulic conductivity field. However, the post-processing yielded very similar average RSE values for the different multiple realizations noisy-GA solutions, as shown in table A.2. This necessitated another approach such as fixing a threshold RSE value and counting the number of samples that pass this threshold. Even this failed as is evident from table A.2 (or figure A.1) for RSE threshold values of 10 and 20. Also, notice the behavior of noisy-GA solutions is similar to the actual case (see figure A.1). This behavior can be attributed to some factors such as: (i) the hydraulic conductivity sample set size being small (100), (ii) during the noisy-GA selection process, average of the RSE values is used for multiple realizations sampling, or (iii) sub optimal GA parameters such as population size or the number of generations. However, it should be noted that complete set of runs could not be carried out because of time constraints and could be an area for future research work.

### A.4 Conclusions

Based on the limited results the noisy-RGA approach may require further pre and post-processing analysis in order to find a robust solution for this test problem. However, some general observations and reasons for the noisy-RGA behavior are given:

- Small variation in hydraulic conductivity fields does not necessarily reflect the same behavior in the concentration profiles generated. It can also result in large variations.
- The variations in the K-fields are disruptive to GA convergence.
- Increasing the number of realizations and averaging the fitness did not improve the GA performance.
- Sampling set size is critical in identifying a robust solution.
- The selection process may influence noisy-GA performance.

**Table A.1 Results obtained for the single source release history reconstruction problem using noisy-RGA approach with multiple realizations and heterogeneous hydraulic conductivity field.**

| Actual Solution | Noisy-GA solutions obtained using different realizations | | | | |
|---|---|---|---|---|---|
| | Number of realizations | | | | |
| (mg/L) | 1 | 2 | 4 | 8 | 16 |
| 70 | 86.88 | 90.16 | 77.62 | 75.02 | 89.14 |
| 50 | 82.73 | 66.22 | 86.79 | 94.6 | 87.06 |
| 90 | 64.61 | 58.26 | 59.82 | 44.02 | 54.74 |
| 40 | 49.9 | 54.49 | 66.27 | 67.65 | 63.84 |
| 50 | 46.75 | 54 | 44.52 | 56.77 | 48.11 |
| 60 | 22.69 | 31.56 | 17.89 | 46.48 | 27.54 |
| 55 | 32.75 | 30.32 | 33.19 | 11.17 | 18.79 |
| 1 | 19.23 | 27.65 | 19.07 | 14.86 | 7.19 |
| 1 | 22.55 | 14.2 | 15.79 | 11.17 | 26.57 |
| 0 | 0.53 | 0.37 | 0.38 | 0.35 | 0.28 |
| **RSE Value** | | | | | |
| **0** | **4.71** | **9.6** | **14.16** | **13.82** | **7.85** |

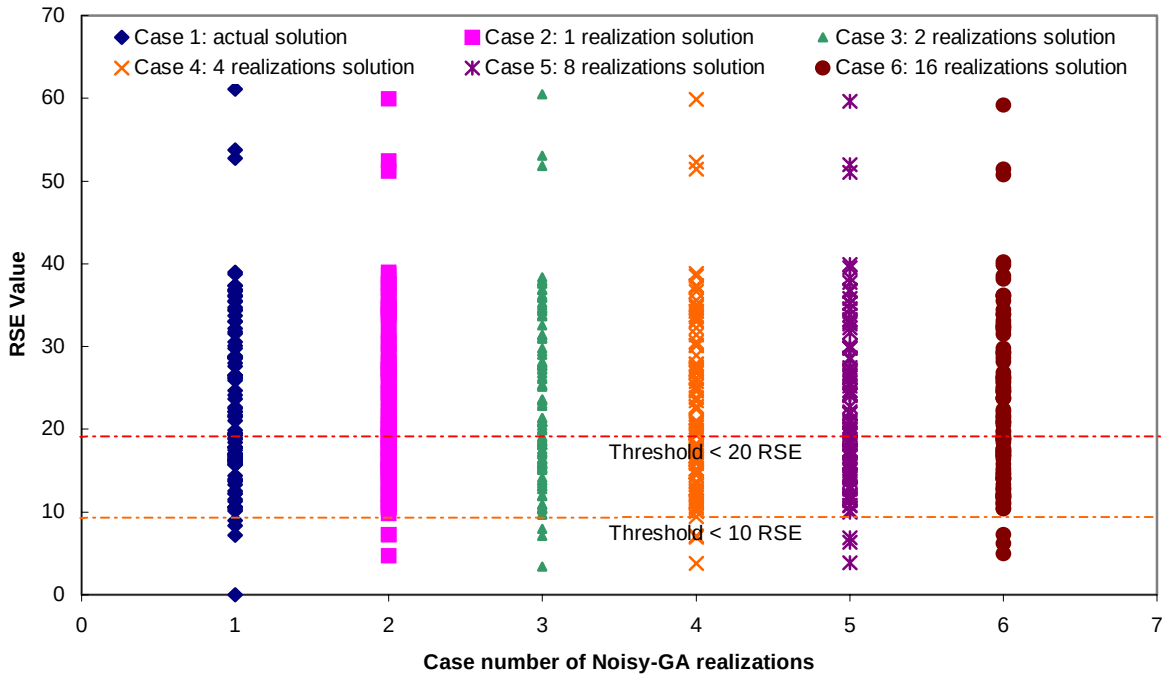



**Figure A.1 Variation in RSE values shown for the different noisy-GA realization solutions using the full hydraulic conductivity sample set (100).**

**Table A.2 RSE values obtained against the full hydraulic conductivity field sample set (100) using the solutions of noisy-RGA and also actual solution.**

| Actual Solution | Number of realizations used in noisy-GA | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| 22.16 | 20.16 | 20.92 | 20.61 | 19.92 | 19.75 |
| 52.73 | 51.22 | 51.81 | 51.46 | 51.08 | 50.75 |
| 36.66 | 35.47 | 36.23 | 35.00 | 34.60 | 33.90 |
| 25.95 | 27.39 | 26.27 | 27.83 | 28.62 | 29.29 |
| 22.12 | 23.67 | 22.83 | 23.44 | 24.20 | 24.56 |
| 16.73 | 18.68 | 18.07 | 17.66 | 18.32 | 18.41 |
| 14.04 | 12.27 | 13.07 | 12.23 | 11.57 | 11.24 |
| 16.93 | 15.31 | 16.13 | 15.14 | 14.46 | 14.02 |
| 22.50 | 20.54 | 21.37 | 20.72 | 20.08 | 19.71 |
| 13.71 | 12.92 | 13.43 | 12.45 | 11.95 | 11.68 |
| 15.73 | 15.08 | 15.20 | 15.15 | 15.16 | 15.22 |
| 16.37 | 16.21 | 15.89 | 16.75 | 16.77 | 17.26 |
| 24.14 | 22.24 | 23.03 | 22.43 | 21.89 | 21.51 |
| 19.39 | 21.57 | 21.00 | 20.27 | 20.93 | 20.86 |
| 35.48 | 33.82 | 34.62 | 33.79 | 33.24 | 32.73 |
| 18.43 | 19.41 | 18.57 | 19.74 | 20.34 | 20.92 |
| 28.77 | 26.85 | 27.67 | 27.03 | 26.46 | 26.04 |
| 16.37 | 14.65 | 15.52 | 14.45 | 13.78 | 13.30 |
| 36.83 | 35.12 | 35.97 | 34.98 | 34.50 | 33.86 |
| 15.98 | 15.86 | 16.30 | 14.87 | 14.47 | 13.98 |
| 61.14 | 59.94 | 60.50 | 59.90 | 59.63 | 59.17 |
| 19.48 | 21.77 | 20.85 | 21.01 | 21.89 | 22.15 |
| 14.39 | 13.29 | 13.74 | 13.31 | 12.80 | 12.77 |
| 23.69 | 23.15 | 23.46 | 22.75 | 22.77 | 22.38 |
| 0.00 | 4.71 | 3.44 | 3.78 | 3.86 | 4.93 |
| 36.04 | 34.47 | 35.29 | 34.23 | 33.80 | 33.13 |
| 19.36 | 17.46 | 18.11 | 17.98 | 17.38 | 17.33 |
| 28.73 | 28.82 | 29.39 | 27.34 | 27.25 | 26.34 |
| 22.69 | 24.37 | 23.59 | 23.86 | 24.68 | 24.88 |
| 17.02 | 17.83 | 17.55 | 17.32 | 17.54 | 17.66 |
| 10.61 | 11.08 | 10.87 | 10.68 | 10.75 | 10.99 |
| 16.34 | 14.68 | 15.35 | 14.81 | 14.30 | 14.08 |
| 26.23 | 24.41 | 25.16 | 24.58 | 24.12 | 23.72 |
| 18.97 | 21.20 | 20.39 | 20.24 | 21.12 | 21.25 |
| 36.19 | 37.61 | 36.80 | 37.39 | 38.32 | 38.52 |
| 38.99 | 37.58 | 38.38 | 37.26 | 36.79 | 36.12 |
| 16.10 | 15.41 | 15.28 | 16.07 | 16.04 | 16.43 |
| 34.53 | 33.85 | 34.51 | 33.03 | 32.75 | 32.01 |
| 21.50 | 20.68 | 21.36 | 19.86 | 19.57 | 18.87 |
| 10.56 | 10.55 | 10.32 | 10.46 | 10.79 | 11.00 |
| 28.01 | 26.36 | 27.21 | 26.20 | 25.61 | 25.05 |
| 8.34 | 10.50 | 9.74 | 9.42 | 10.10 | 10.34 |
| 53.75 | 52.43 | 53.09 | 52.31 | 51.96 | 51.42 |
| 16.21 | 16.09 | 15.82 | 16.40 | 16.60 | 16.93 |
| 10.39 | 11.44 | 10.81 | 11.17 | 11.68 | 12.07 |
| 28.66 | 26.73 | 27.60 | 26.84 | 26.21 | 25.75 |
| 17.82 | 16.39 | 17.17 | 15.98 | 15.56 | 14.96 |
| 8.98 | 7.24 | 7.99 | 7.10 | 6.38 | 6.19 |

| | | | | | |
|---|---|---|---|---|---|
| 24.68 | 22.66 | 23.55 | 22.77 | 22.14 | 21.66 |
| 7.17 | 7.25 | 7.09 | 6.93 | 6.89 | 7.26 |
| 35.45 | 34.15 | 34.98 | 33.61 | 33.20 | 32.42 |
| 33.01 | 34.51 | 33.65 | 34.26 | 35.28 | 35.48 |
| 30.55 | 32.01 | 31.39 | 31.41 | 32.23 | 32.26 |
| 27.65 | 25.86 | 26.74 | 25.77 | 25.18 | 24.62 |
| 36.90 | 38.58 | 37.57 | 38.51 | 39.49 | 39.86 |
| 16.84 | 16.61 | 16.40 | 16.88 | 17.12 | 17.37 |
| 17.81 | 19.00 | 18.08 | 19.26 | 19.92 | 20.51 |
| 12.51 | 13.36 | 13.12 | 12.60 | 12.91 | 12.93 |
| 31.79 | 30.60 | 31.35 | 30.10 | 29.75 | 29.05 |
| 28.89 | 27.19 | 27.98 | 27.20 | 26.62 | 26.15 |
| 30.12 | 29.23 | 29.75 | 28.90 | 28.63 | 28.15 |
| 26.49 | 24.75 | 25.35 | 25.23 | 24.68 | 24.58 |
| 31.55 | 30.31 | 30.93 | 30.13 | 29.82 | 29.33 |
| 34.36 | 33.71 | 34.23 | 33.19 | 32.93 | 32.43 |
| 12.21 | 12.75 | 12.72 | 11.78 | 12.06 | 11.84 |
| 21.03 | 19.47 | 20.03 | 19.85 | 19.29 | 19.25 |
| 29.80 | 28.12 | 28.97 | 27.95 | 27.42 | 26.83 |
| 10.56 | 11.03 | 10.48 | 11.20 | 11.53 | 12.05 |
| 38.73 | 37.04 | 37.79 | 37.14 | 36.63 | 36.19 |
| 16.62 | 16.21 | 16.38 | 15.89 | 15.96 | 15.76 |
| 16.19 | 15.16 | 15.74 | 14.87 | 14.27 | 13.98 |
| 26.56 | 25.27 | 26.04 | 24.84 | 24.38 | 23.74 |
| 15.79 | 14.72 | 15.05 | 14.96 | 14.56 | 14.64 |
| 21.61 | 20.57 | 20.80 | 20.93 | 20.77 | 20.79 |
| 17.82 | 17.63 | 17.22 | 18.26 | 18.52 | 19.00 |
| 21.73 | 20.03 | 20.88 | 19.84 | 19.34 | 18.78 |
| 37.39 | 36.12 | 36.93 | 35.62 | 35.18 | 34.44 |
| 37.39 | 38.96 | 38.00 | 38.85 | 39.92 | 40.22 |
| 33.75 | 31.88 | 32.59 | 32.26 | 31.72 | 31.45 |
| 32.17 | 30.25 | 30.99 | 30.59 | 30.09 | 29.76 |
| 13.32 | 14.23 | 13.87 | 13.66 | 14.08 | 14.22 |
| 11.32 | 10.67 | 10.45 | 11.28 | 11.34 | 11.82 |
| 17.09 | 19.25 | 18.60 | 18.09 | 18.79 | 18.81 |
| 17.84 | 18.94 | 19.10 | 17.16 | 17.40 | 16.71 |
| 26.12 | 28.12 | 27.24 | 27.52 | 28.44 | 28.63 |
| 13.81 | 13.92 | 14.12 | 12.88 | 13.16 | 12.68 |
| 18.82 | 17.75 | 18.30 | 17.49 | 17.16 | 16.81 |
| 11.52 | 12.22 | 12.02 | 11.41 | 11.81 | 11.74 |
| 19.23 | 19.58 | 19.89 | 18.23 | 18.32 | 17.65 |
| 10.16 | 9.82 | 9.67 | 10.09 | 9.98 | 10.45 |
| 32.99 | 34.86 | 33.78 | 34.70 | 35.80 | 36.15 |
| 28.65 | 27.31 | 28.14 | 26.83 | 26.33 | 25.63 |
| 26.43 | 24.47 | 25.28 | 24.74 | 24.09 | 23.76 |
| 34.74 | 37.03 | 35.87 | 36.68 | 37.74 | 38.11 |
| 19.90 | 17.98 | 18.88 | 17.98 | 17.30 | 16.84 |
| 17.52 | 17.31 | 16.92 | 17.97 | 18.17 | 18.70 |
| 15.76 | 15.54 | 15.81 | 14.78 | 14.92 | 14.51 |
| 19.59 | 17.88 | 18.60 | 18.03 | 17.43 | 17.16 |
| 28.56 | 27.26 | 27.87 | 27.16 | 26.83 | 26.41 |
| 12.24 | 11.95 | 11.90 | 12.05 | 11.96 | 12.25 |
| 22.16 | 20.16 | 20.92 | 20.61 | 19.92 | 19.75 |

| | | | | | |
|---|---|---|---|---|---|
| **Average RSE values** | | | | | |
| **23.26** | **22.90** | **23.07** | **22.68** | **22.62** | **22.47** |
| **Number of RSE values < 10** | | | | | |
| **4** | **4** | **5** | **4** | **4** | **3** |
| **Number of RSE values < 20** | | | | | |
| **49** | **47** | **46** | **49** | **48** | **48** |

# Appendix - B

# Preliminary Investigation of Modeling to Generate Alternatives (MGA)

Inverse problems are generally ill posed i.e. they suffer from non-uniqueness (multiple solutions), non-existence (no solution) and instability (small error in the measured observations results in large variation in parameter estimates) of the solution. To address the "non-uniqueness" issue, a small set of alternative solutions that are far apart in the decision space but close in the objective function space can be obtained using the MGA technique. For example, in groundwater modeling two different source locations can produce same observation data (i.e. concentration release histories etc). Each alternative solution obtained by GA can be further fine-tuned using local search methods (multi-point local search). From the small set of alternatives a single best solution can be selected based on additional constraints or human judgment. Special GA operators based on niching by sharing or crowding mechanisms have been developed by [Loughlin et al. 2001] for the MGA technique.

## B.1 MGA approach

The initial MGA implementation adopted in this work is very similar to the approach of Loughlin et al. At the end of each GA generation a small set of MGA alternatives is selected from a *"candidate pool"* that is within some cutoff range of the best in the population (e.g. within 15% of the best in objective space). The first MGA alternative is the best performing GA individual. The second alternative is selected from the candidate pool such that it is farthest from the first in decision space based on a Euclidean distance metric to be described later in the following section.

The remaining alternatives are selected such that they are farthest from all previous alternatives. These selected MGA candidates are assigned higher fitness values so that they are more likely to be carried over to the next generation. This ensures that the MGA solutions have the unfair advantage of getting selected during the selection process. Preliminary test results for the 3D source reconstruction problem were not encouraging,

probably due to absence of restricted mating and subpopulation concepts in this implementation. A modified implementation was adopted for further investigation incorporating the following: (1) restricted mating and co-evolutionary strategies by maintaining subpopulations (2) Varying the size of the candidate pool with the generation number, i.e. by having a higher cutoff value initially and then reducing it with the generation index, and (3) elitism for MGA alternatives. The effectiveness of this approach was initially tested with a two-dimensional multi-modal test problem and then applied to our groundwater inverse problems. The steps involved in the new MGA approach are as follows:

Step 1. Select the candidate pool for MGA alternatives within certain range of the best in the population

gen_factor = 0.15 +0.35*(maxgen-gen)/maxgen

rsemin ≤ rse_value(candidates) ≤ (1+gen_factor)*rsemin

Where, gen = current generation number and maxgen = maximum number of generations.

If the candidate pool size is larger than the number of MGA alternatives desired, then go to next step; otherwise do nothing for the MGA process. It is hoped that since the gen_factor will be large (~0.5) in the beginning many individuals will qualify for the candidate pool and the candidate pool will be larger than the alternatives.

Step 2. The next step is to select the MGA alternatives. The first MGA alternative is the best individual in the population. The other alternatives are selected from the candidate pool based on the distance metric given below, so that they are farthest from all previous alternatives.

Distance, $r = \sqrt{\sum_{i=1}^{n} (x_1 - y_1)^2 + (x_2 - y_2)^2 + ..... + (x_i - y_i)^2}$

Where, n = number of decision variables

$\underline{x} = \{x_1, x_2, ....., x_n\}$

$\underline{y} = \{y_1, y_2, ....., y_n\}$

$\underline{x}$ and $\underline{y}$ are the two alternate solutions.

When identifying more than two MGA solutions and for finding the 3rd or later alternatives, the minimum of the distances from already selected alternatives is calculated. The candidate solution with the maximum of the minimum distance from the other alternatives is selected as the next alternative. This can be can be represented by the following expression.

$$MGA_{3,\ldots,m} = \max_{i=3,..,m}\left(\min_{\substack{j=1,..,p \\ k=1,..,q}}(r_{ijk})\right)$$

where

$m$ = number of MGA solutions desired

$p$ = candidate pool size

$q$ = number of MGA solutions already known ( $\geq 2$)

$r$ = distance (computed as shown above)

Step 3. Divide the population in to subpopulations equal to the desired number of MGA alternatives. If more than one MGA alternative belongs to a subpopulation, retain the best for that subpopulation and exchange the remaining alternatives with similar individuals (based on distance metric) in subpopulations without any MGA alternatives. With this approach every subpopulation will be seeded with exactly one MGA alternative.

Step 4. Once the MGA alternatives are properly seeded, reproduction is restricted to subpopulations.

Step 5. Several approaches are available for selection. Some of them were implemented and experimented, and are given below:

- In this approach fitness scaling is performed before selection. The individual fitness is a combination of fitness and distance from the MGA alternative in that subpopulation.

$fitness_i = w_f * fit_i + w_d * dist_i$

where

$w_f$ and $w_d$ are weight factors for fitness and distance.

$fit_i =$ fitness of individual *i*

$dist_i =$ Euclidean distance between between MGA alternative and individual *i*.

Note the weighting factors were chosen such that slightly more preference is given to solutions with better fitness.

- In this second approach a record of the number of fit individuals in each of the subpopulations is maintained. Here the fit corresponds to the threshold fitness used in selecting the candidate population in step 1. If more than 50% of individuals in the subpopulation are fit, then a flag is set to identify the subpopulation. This criteria is used during the selection process, so that the populations are balanced and do not prematurely converge to an inferior solution. During the selection process (tournament selection) the individuals in the first subpopulation, which has the best performing individual (based on fitness), are selected solely based on fitness. For other subpopulations the selection can be based on fitness and/or distance metric approaches as described below:
    - In this approach the centroid of the subpopulations is used in the calculation of the distance metric. The centroid is simply the average of all the individuals in the subpopulation and can be represented by the following expression.

      $$\overline{X_j} = \sum_{i=1,..,m} X_{i,j} \Big/ m \quad , \; j = 1,..,\text{n}$$

      Where,

      $m$ = subpopulation size

      $n$ = number of decision variables

      $X =$ vector of decision variables

      The distance metric is now the distance (Euclidian) between the centroid of a subpopulation and the individual and is calculated using a similar expression given in step 2.

**(1)** If less than 50% of the population is fit then the selection is based on fitness only.

**(2)** If the subpopulation has more than 50% fit individuals, then the selection between two fit individuals is based on the distance metric only. If one of the individual is fit and the other below the fitness threshold then the selection is based on fitness alone and the fitter individual is selected. If both the individuals selected are below the fitness threshold, then also the selection is based on fitness alone.

- In this approach, instead of using the centroid of the subpopulation, the distance metric is computed from individual MGA alternatives in those subpopulations. The distance (Euclidean) is computed between the MGA alternative and a individual of the subpopulation using an expression similar to the one given in step 2.

However, the approach using the centroid of the population worked better for the test problem and is used for any further simulations using MGA's.

Step 6. The MGA process in steps 1-5 is performed for every GA generation.

**B.2 Test problem**

As mentioned earlier the MGA implementation is evaluated with a test problem where the alternatives are known. The two dimensional multi-modal problem tries to maximize the following function:

$$F(x, y) = \sin(19\pi x) + 1.7/x + \sin(19\pi y) + 1.7/y + 2$$

This function has the peak value of 5.15 at location (0.974, 0.974) and is the global optimum. The other alternative solutions in the decision space that are maximally different in the decision space but are reasonably good in the objective space are (0.018, 0.974) (0.974, 0.018) and (0.45, 0.55) or (0.55, 0.45). Additional information and solutions to this multi-modal problem are reported in [Loughlin et al. 2001]. The results obtained by using our implementation for this test problem are discussed in the next section.

Once the MGA implementation was validated based on the results, it is used for the real 3D three sources release history reconstruction problem using heterogeneous flow field. The problem setup is similar to the source release history reconstruction problem described earlier in section 5.4. The results for this problem are discussed in the next section.

**B.3 Results**

For the 2D test function problem, an initial GA population size of 100 is used and other GA parameters are assumed. Four MGA alternatives are desired and the population is divided (virtually) in to four subpopulations that undergo restricted mating as described earlier. The MGA procedure described earlier is adopted. The GA+MGA process is run for 50 generations and with different initial random seeds. Figure B.1 shows the distribution of the solutions obtained for 50 different runs. The mean distance of MGA alternatives (based on 50 runs) is computed as 0.75. This suggested the implementation is working.

Results for the three sources release history reconstruction problem, using the MGA approach with two different selection approaches, one based on fitness only and the other based on psuedo fitness (objective function value and distance metric) are shown in figures B.2 (a) and (b) and B.3 (a) and (b). These figures show that the selection based on psuedo-fitness produced a very different $2^{nd}$ MGA alternative compared to the fitness only approach. However, other remaining two alternatives 3rd and $4^{th}$ were very similar for both selection approaches. The primary reason for this behavior can be attributed to the lack of non-uniqueness nature of the release history problems.

Preliminary tests carried out on 2D problems indicate that the source reconstruction problems (location and concentrations) exhibit more non-uniqueness properties than release history problems. Therefore, the MGA approach is more suited for these problems. This research could not be carried further for time constraints and can be a potential topic for future research.

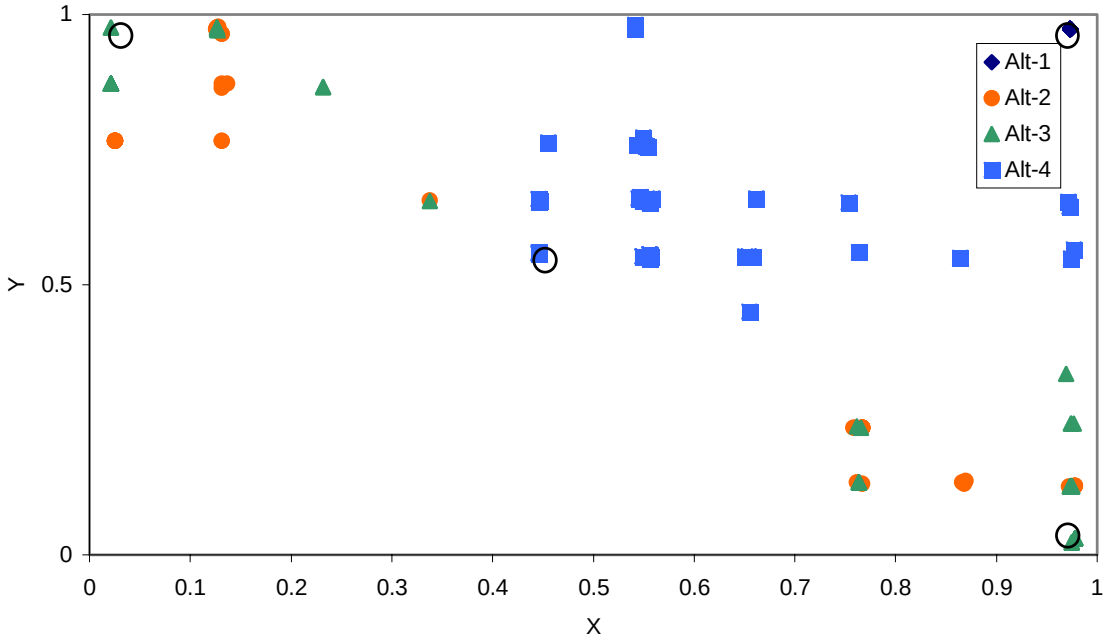
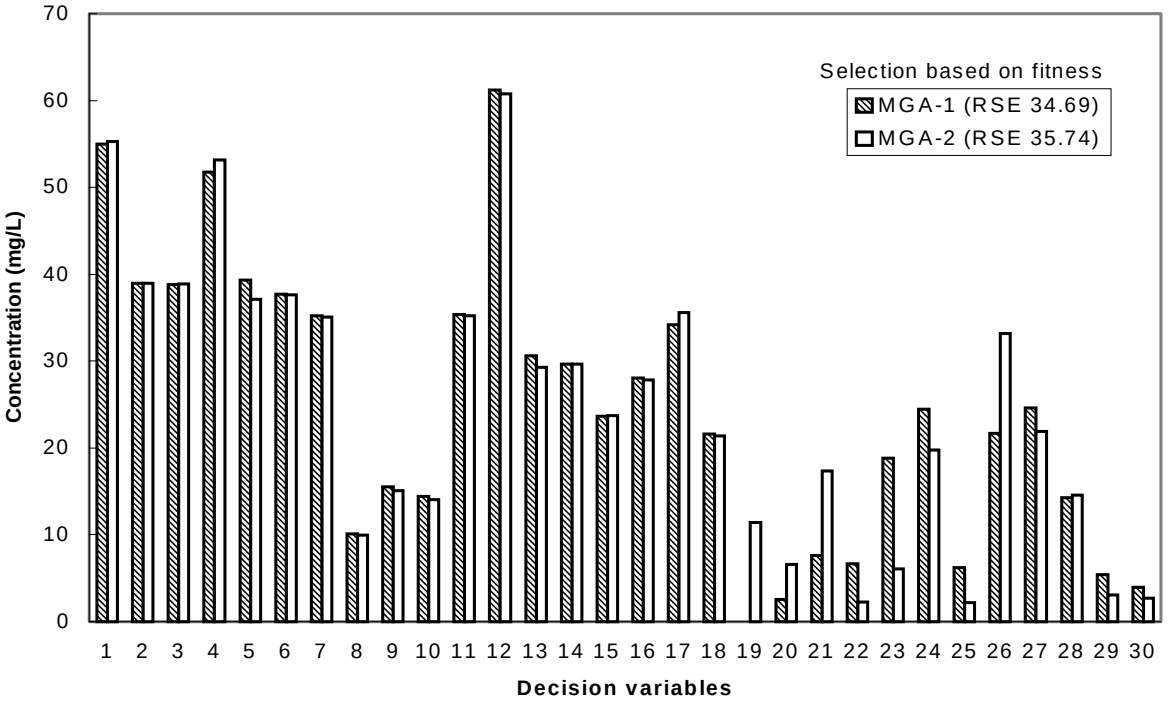
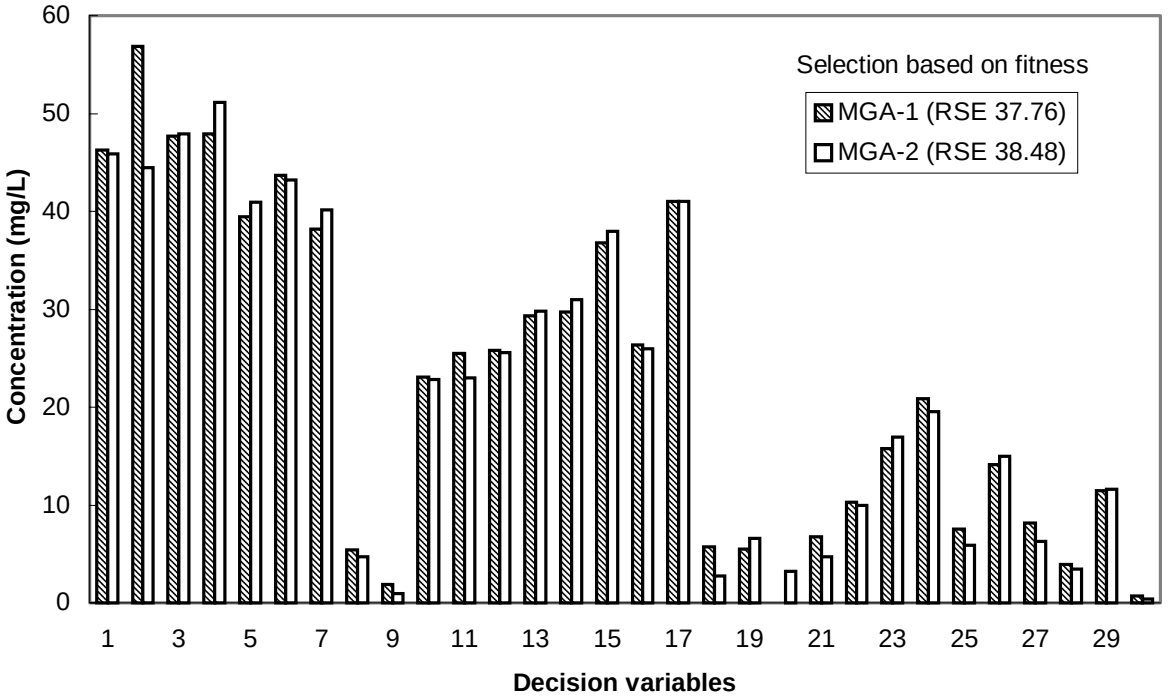



**Figure B.1 Distribution of MGA solutions for 50 runs. Solutions marked with circles are the MGA solutions.**

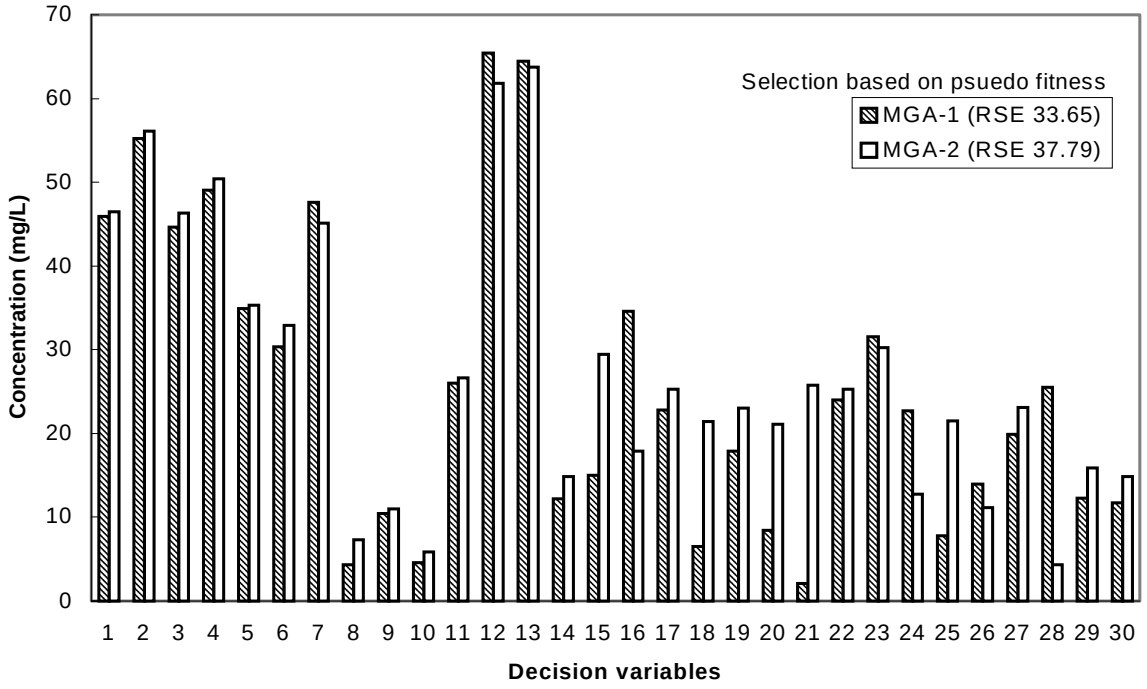



**(a)**

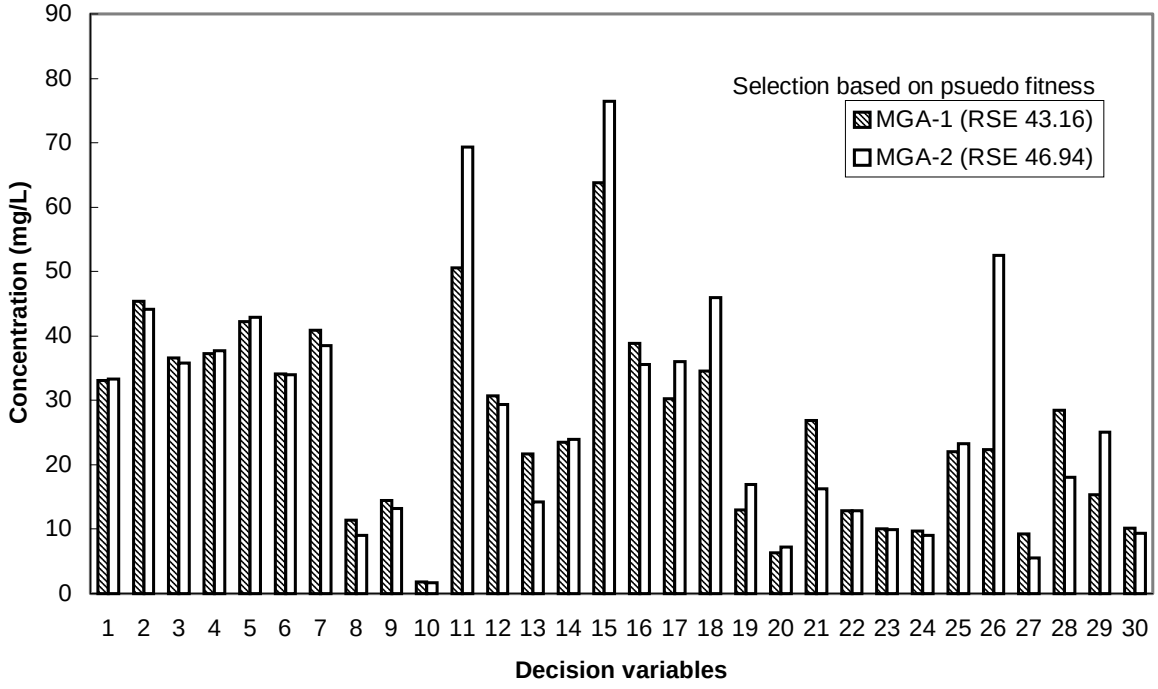



**(b)**

**Figures B.2 (a) and (b) Difference between MGA alternatives 1 and 2 for selection based on fitness for two different runs.**




**(a)**




**(b)**

**Figures B.3 (a) and (b) Difference between MGA alternatives 1 and 2 for selection based on psuedo fitness for two different runs.**