A Multiscale Master-Slave Parallel Genetic Algorithm with Application to Groundwater Remediation Design

Meghna Babbar

Dept. of Civil and Environmental Engineering
University of Illinois
Urbana, IL 61801
mbabbar@uiuc.edu

Barbara S. Minsker

Dept. of Civil and Environmental Engineering
University of Illinois
Urbana, IL 61801
minsker@uiuc.edu

Abstract

Numerical models are an integral part of many optimal engineering design problems. The accuracy of fitness function evaluations for these applications depends on the spatial grid sizes used in the numerical models. Fine grids usually improve the accuracy of the solutions, but they can also pose major bottlenecks in the computational efficiency of the algorithms. The need to carefully select a grid size that can maintain the numerical accuracy of the solutions, along with being computationally less exhaustive, is very crucial for applications that use genetic algorithms. In this work, we present a multiscale parallel genetic algorithm that can be used to improve the performance of engineering design problems that use spatial grids. The algorithm's efficacy is tested using a groundwater remediation design case study.

1 INTRODUCTION

Many engineering design problems can be solved using optimization techniques coupled with simulation models to identify effective designs. For example, in the groundwater remediation field genetic algorithms have been used extensively with numerical simulation models to analyze the performance of candidate remediation designs [Ritzel et al., 1994, Smalley et al., 2000, Wang, 1991, Wang and Zheng, 1997, Aksoy and Culver, 2000, Gopalakrishnan et al., 2001]. The degree of accuracy of these numerical models is a function of many factors, including grid size. Finer grids usually improve accuracy, but can increase computational effort substantially.

Some work has been done in the past to study the effect of multiscale grid approximations on optimization models [see *Albert and Goldberg*, 2001, *Liu and Minsker*, 2001]. *Albert et al* used a simple genetic algorithm (SGA) to explore the use of coarse grid approximations for solving a numerical integration problem. *Liu et al* (2001) devised a full multiscale approach for in-situ bioremediation design using a derivative-based optimization method called SALQR.

In this paper, we explore the potential for a new multiscale parallel approach to improve the performance of a groundwater remediation design optimization problem described below.

2 GROUNDWATER REMEDIATION PROBLEM AND OBJECTIVES

This study examines a groundwater remediation design problem whose two major components are the numerical model and the optimization model. The main objective of the design problem is to identify the most cost effective groundwater treatment strategy that keeps the human health risk due to contamination within a specified limit. The numerical model simulates the flow of groundwater and transport of contaminants for each candidate remediation plan. The optimization model risks evaluates the objectives and constraints and identifies optimal designs using a simple genetic algorithm (SGA). Below is a description of the two models.

2.1 NUMERICAL MODEL

The groundwater management model was based on a confined, heterogeneous, isotropic aquifer with a BTEX spill, studied previously by *Smalley et al* (2000). The modeled aquifer is 480m long, 240m wide and 20m deep. Groundwater Modeling System (GMS) modules MODFLOW (*McDonald et al*, 1988) and RT3D (*Clement et al*, 1998) were used to create and run the numerical

model. The model assumed steady state flow of groundwater in the aquifer. In order to study solely the effect of noise due to grid size (which is the main objective of this research), the uncertainty in hydraulic conductivities and risk parameters (*Smalley et al*, 2000 and *Gopalakrishnan et al*, 2002) were assumed to be absent. Hence, only a single realization of the generated hydraulic conductivity fields was used in this work. The aquifer was assumed to have constant head boundaries at the two ends of its 480m length, with a mean hydraulic gradient of 0.00146. The porosity of the field was assumed to be 0.3.

The initial plume was created using 10 contaminant sources, with contaminant concentrations varying from 50mg/l to 600mg/l, on a grid size of '144 by 72'. The numerical model was run for a stress period of 2 years to create the initial spatial distribution of the plume. The longitudinal dispersivity was assumed to be 15 m, with a ratio of transverse and longitudinal dispersivity of 0.2, and ratio of vertical and longitudinal dispersivity of 1.0. The aquifer was assumed to have a soil bulk density of 2000 kg/m^3 and a linear sorption reaction with reaction constant of 0.000062 m^3/kg. Once the initial plume was created, the concentrations and hydraulic conductivities were spatially averaged to grid sizes of '16 by 8' and '48 by 24' elements, each having one layer. These averaged concentrations were then used as the starting concentrations for the plume on the coarser grids in the management model. See Figures 1 and 2 for the initial plume concentration distributions on the two grid sizes.

For the remediation model (see Figure 3), the only source of contamination that existed in the field was assumed to be the initial distribution derived in the previous step. Fifteen monitoring wells were used to observe the concentrations of the plume in the aquifer for a 20-year remediation period. Pump and treat technology was used for the remediation process, with a maximum of three remediation wells available for extraction/injection at the 58 candidate locations shown in Figure 3. Each well was assumed to have a maximum pumping capacity of 250 m^3/day. Since most BTEX compounds are semi-volatile, an ex-situ air stripping technology was selected for the extracted water.

Once the concentration within the plume was obtained from the numerical model, a human health risk assessment model was used to predict human health risks at an assumed exposure point 200 m offsite. See *Smalley et al* (2000) for more details on the risk assessment model.

2.2 OPTIMIZATION MODEL

The objective of the remediation design problem is to minimize total treatment cost such that constraints for risk, drawdown and pumping rates are met. Total treatment cost consists of remediation well costs, monitoring costs and ex-situ treatment costs [see *Gopalakrishnan et al*, in press 2002, for details]:

$$\textbf{\textit{Min}}\ C_{TOT} = C_{REM} + C_{MON} + C_{SYST} \qquad (1)$$

The constraints for the problem are:

1) Total individual lifetime human health risk, $Risk^{Total}_{t,k}$, at any time t and exposure location k should be less than the target level of risk, *TR*.

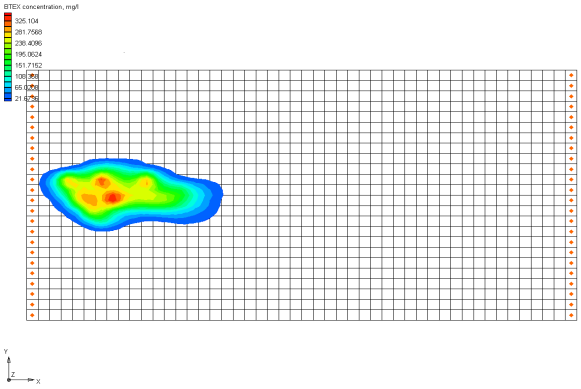

Figure 1. BTEX plume on '48 by 24' grid size

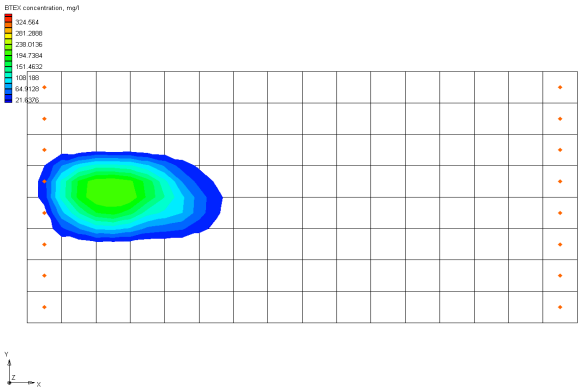

Figure 2. BTEX plume on '16 by 8' grid size

$$Risk^{Total}_{t,k} = Risk^{w}_{t,k} + Risk^{shw}_{t,k} + Risk^{nc}_{t,k} \le TR \quad \forall t, \forall k \qquad (2)$$

where,

$Risk^{w}_{t,k}$, is cancer risk due to injestion of contaminated drinking water,

$Risk^{shw}_{t,k}$, is cancer risk due to inhalation of volatile compounds emitted from contaminated water while showering,

$Risk^{nc}_{t,k}$, is cancer risk due to inhalation of volatiles through other non consumptive routes.

See *Smalley et al.* (2000) for details on how the risk is calculated.

2) The pumping rates of the wells, u_i, should lie within the maximum (u_{max}), and minimum (u_{min}), capacities for any remediation well, *i*.

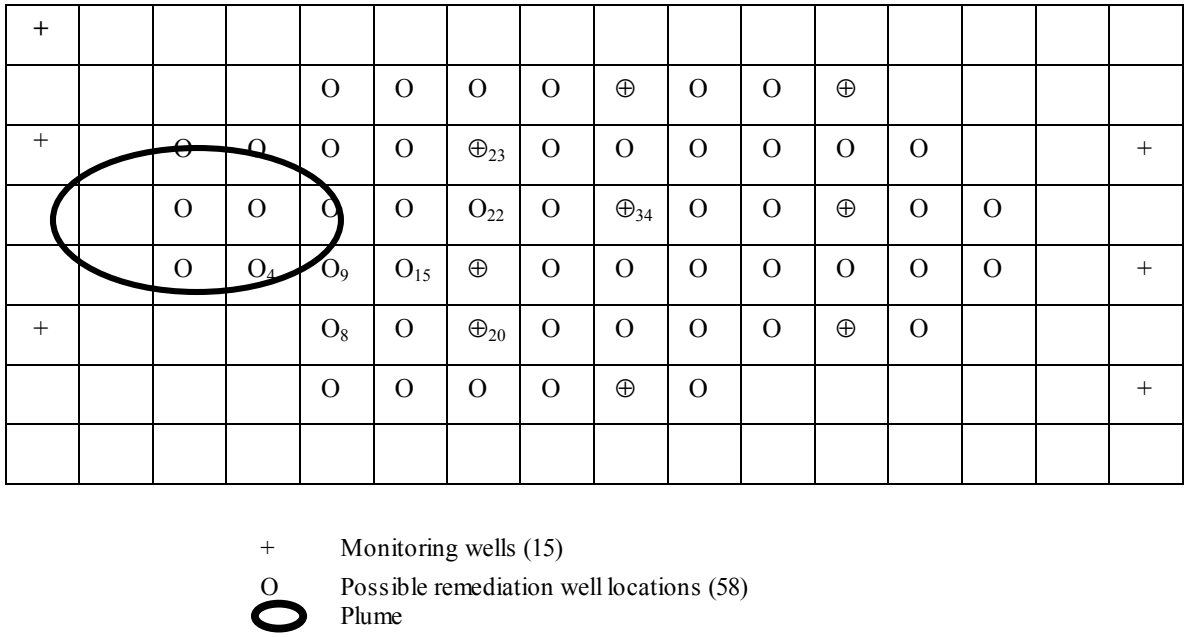

Figure 3. Plan View of the Case Study Aquifer (grid size 16 by 8)

$$u_{min,i} \le | u_i | \le u_{max,i} \qquad \forall i \tag{3}$$

1) The hydraulic head, $h_{i,l}$, for a remediation well should lie within the maximum ($h_{max,l}$) and minimum ($h_{min,l}$) heads allowed at each well location, l.

$$h_{min,l} \le h_{i,l} \le h_{max,l} \qquad \forall i \quad \text{at each } l \tag{4}$$

For the genetic algorithm application, decision variables for the remediation design are well locations and pumping rates for three remediation wells, a variable to decide whether the wells perform extraction or injection, and a variable to determine whether the wells are installed or not. These decision variables are encoded into a binary string with 48 bits. Constraints for pumping rates (equation 3) are enforced by limiting the number of bits used to represent the pumping rate decision variables in the genetic algorithm. The head and risk constraints in equations 2 and 4 are added to the total treatment cost, using a linear penalty for violations, to create the overall fitness function.

$$\textit{Fitness function} = - C_{TOT} - w_1 * \textit{Risk violation} - w_2 * \textit{head violation}. \tag{5}$$

The penalty weights for risk (w_1) and head (w_2) violations were each set to 1000. Penalties are based on violations with respect to a risk limit of 10^{-5} (or a 1 in 100,000 increased lifetime cancer risk), and an allowable maximum drawdown of 0.12 m.

3 SIMPLE GENETIC ALGORITHM DESIGN METHODOLOGY

Simple genetic algorithms (SGA) are good optimization tools for groundwater remediation design problems, which are discrete, non-convex, and discontinuous. Reed et al. (2000) have shown that effective selection of various GA parameters is important to ensure that the GA converges to optimal or near optimal solutions for practical engineering problems. In their work, they also showed that theory from the current GA literature can be used in the GA design methodologies for identifying realistic parameter values. A brief overview of their suggestions that we have used in our work is given below.

3.1 POPULATION SIZING AND PARAMETER SETTINGS

The building block theory says that GAs find good solutions by assembling building blocks, which are high quality "pieces" of chromosomes. Goldberg et al. (1992) have emphasized the importance of GA population sizing to ensure a sufficient initial supply of building blocks. The population size can be estimated using the equation developed by *Harik et al* [1997] which is based on a random walk model for the initial supply of building blocks (BBs) and the gambler's ruin model for representing the interaction of competing BBs:

$$N \ge -2^{k-1} \ln(\alpha) \frac{\sigma_{bb} \sqrt{\pi (m-1)}}{d} \tag{6}$$

where,

N = population size,

k = BB order,

α = probability of failure (less than 5%),

d = signal difference,

σ_{bb} = standard deviation or noise interference between competing BBs,

In this equation, the term '$\sigma_{bb}\sqrt{\pi(m-1)}$' can be approximated as the standard deviation of fitness σ_f for a large initial population (*Reed et al*, 2000). For the '16 by 8 grid' size problem and '48 by 24 grid' size problem, fitness evaluations were done for an initial population of 1000 random individuals. Using the fitness of this population, the standard deviation 'σ_f' was estimated to be 1,472,000 for the '16 by 8 grid' problem and 1,731,000 for the '48 by 24 grid' problem.

The signal difference 'd', which represents the difference in fitness between best and second best building blocks in the population, is unknown in real-world applications. We estimate the signal difference using a probabilistic approach that is based on the statistical frequency analysis theory (*Hogg et al, 1978*). This approach sorts the fitness evaluations for the initial random population and organizes them into classes that are probabilistically distinct. Based on this frequency analysis theory, the number of bins created is approximately equal to $\sqrt{n}$, where n is the total number of individuals (1000 in this case). The number of individuals per bin is evaluated based on the condition that np_i should be ≥ 5, where p_i is the percentage of total number of individuals in that bin. See *Hogg et al* for more details on this theory. Once the histograms are created, the signal difference is estimated to be the difference in fitness of the first two bins (i.e. second lowest and lowest fitness in this case, where lower fitness is better). Signal difference for this problem was estimated to be 157,821 for the '16 by 8 grid' and 198,541 for the '48 by 24 grid'. Figure 4 and Figure 5 show the frequency analysis histograms for the '16 by 8' and '48 by 24' cases.

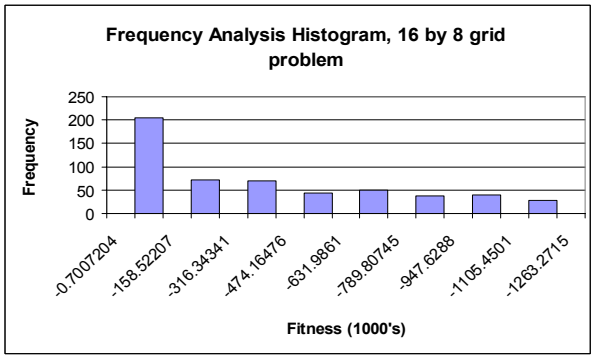

Figure 4. Histogram of fitness for initial population, 16 by 8 grid problem

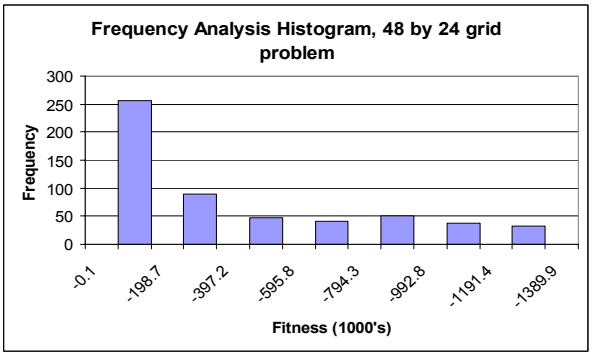

Figure 5. Histogram of fitness for initial population, 48 by 24 grid problem

Since we cannot identify the actual building block order 'k' in equation 6 for real-world problems, the population sizes for both grid problems are calculated using the above parameters for k=1, k=2, k=3, and k=4. Higher building block orders are usually disrupted by the crossover operation (*Reed et al*., 2000) and hence are not considered. Table 1 shows the population sizes estimated for different building block orders.

Table 1. Population sizes for the different building block orders

Building Block order, k	Population size, '16by8' grid problem	Population size, '48by24' grid problem
1	30	25
2	60	50
3	120	100
4	240	200

Tournament selection with a tournament size, 's', equal to 4 was used for this problem. Per Reed et al. (2000), probability of crossover (P_c) was estimated to be approximately '(s-1)/s' or 0.75 in this case. Probability of mutation (P_m), is set equal to 1/N, where N is the population size (*De Jong et al*., 1975). For tournament selection, convergence time for the SGA is estimated to be less than twice the chromosome length (*Thierens et al*., 1998).

The $\mu + \lambda$ selection scheme, without replacement, was used for selecting the best individuals for successive generations, where μ represents the child population and λ represents the parent population. In this selection scheme the parent and child population are combined and the best individuals from the joint population are selected as the parent population for the next generation. In the 'without replacement' selection condition, any individual selected is not placed in the contest again. This selection scheme was found to be more effective for this problem than a policy with replacement.

An 'injection' method (adapted from work done by *Reed*, 2002) proved to be very effective in solving the simple

GA problem for different building block orders. The SGA was first solved for building block order 'k=1', which is the smallest population size. The best individual found in this problem was then injected into the starting population of the next order (i.e. 'k=2') problem. This method of injecting best individuals from previous 'building block order' problems into the starting random population of the next higher 'building block order' problem was repeated successively for higher order problems. Using this approach, we insure that the search always moves towards improving the solution quality in sequential runs. The population size was sequentially increased in this manner until more than 90% of the population converged to the same or sufficiently similar solutions.

3.2 SGA SOLUTIONS

The SGA was used initially to solve the remediation problem for both '16 by 8 grid' and '48 by 24 grid' sizes. Tables 2 and 3 depict the solution qualities for the different grid problems. The fine and coarse grid problems identified different solutions due to the reduced accuracy of the coarse grid.

Building block order 'k=3' was selected as the most suitable population size for all subsequent analysis. The time spent on computation for 'k=3' is much lesser than that for 'k=4', but the solution quality varies less than 2% from the best solution identified in each case.

4 MULTISCALE MASTER-SLAVE PARALLEL GENETIC ALGORITHM

Evaluating fitness functions with numerical models is often computationally intensive, with each fitness function evaluation requiring a large amount of processor time. The simplest way to address this problem is a master slave parallelization, in which the 'master' processor distributes the fitness function evaluations among several 'slave' processors (*Cantu'-Paz et al*, 1999). The slave processors evaluate the fitness function and return the information back to the master processor, who does all of the GA operations (see Figure 6). Master-slave GAs search the decision space in the same way as an SGA, but can be solved more quickly.

Table 2. Simple GA solution for '16 by 8 grid' problem

Building block order	Population size	Best fitness, (1000's)	Convergence time (# of generations)
1	30	-185.7	62
2	60	-185.6	42
3	120	-184.4	67
4	240	-183.3	96

Table 3. Simple GA solution for '48 by 24 grid' problem

Building block order	Population size	Best fitness, (1000's)	Convergence time (# of generations)
1	25	-183.9	96
2	50	-183.9	45
3	100	-182.1	58
4	200	-180.6	51

The two most important limiting components of the master-slave GA are the computational time for each fitness evaluation and the communication time between the master and slave processors. Since communication between the master and each slave occurs every generation, the cost of communicating individuals creates a tradeoff between computation and communication costs. Therefore, master-slave GAs are more effective for problems that have time-intensive fitness function evaluations. Theoretically, *n* processors should be able to give a speedup of *n* in a master-slave parallel setup. However in real life, various factors like problem characteristics, processor characteristics, etc. limit the actual speedups obtained. *Grefenstette* (1995) found that a complex robot learning task gave him 80% efficiency, versus 50% efficiency for an easier task.

Since this work addresses the problem of using a fitness function that implements computationally intensive numerical models, the master-slave setup is a practical tool for decreasing the computational time. For our experiments, 128 dedicated processors on the SGI Origin 2000 at the National Center for Supercomputing Applications (NCSA) were used.

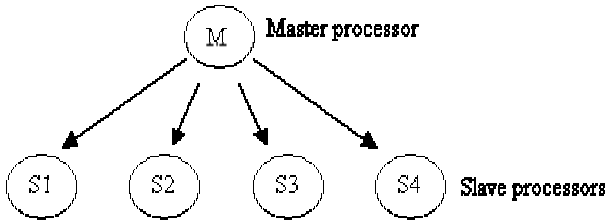

Figure 6. Master and Slave parallel processors

The master-slave approach was initially used to run the '48 by 24' (fine) grid case alone. A simple multiscale approach was then investigated, in which the '16 by 8' (coarse) grid was used initially and then, at a later stage during the GA run, the problem shifted to the '48 by 24' (fine) grid case. In our experiments we tested various possible stages during the run when the shift to the finer grid could be done. We also observed the effects of the grid shifts on noise. The GA creates noise in the

population while evaluating building block fitness (σ_f) because a finite population size is used. The noise or uncertainty that exists in the fitness function evaluations because of the grid approximations can also add to the noise of the population when the grid size changes. In this work we observe this phenomenon by tracking the variance of the population every generation, which is how noise is measured. Also, from our initial results with the SGA, it is clear that the coarse grid problem and fine grid problem tend to drive the SGA towards somewhat different solutions (see Tables 2 and 3). However, running the GA initially on a coarse grid problem should drive the solution close to the optimal solution. When the fine grid problem takes over the search, it should then refine the solution to identify a more accurate optima.

The optimal population size identified previously for the coarse grid problem, which is 120 (for 'k=3'), is larger than that for the fine case (i.e. 100). For the multiscale master-slave problem a population size of '120' was selected to ensure an adequate population for both grid sizes. To maximize the resource usage on the 128-processor parallel computer, the population size was then increased to 124 so that each processor works with at least one individual at a time. To implement a multiscale version of the injection approach, the best individual estimated from the SGA runs for building block order 'k=2' (from the '16 by 8 grid' problem) was inserted into the initial 'k=3' population for the multiscale master-slave GA at generation 0.

Four cases were created to investigate the effects of multiple scales on the optimization algorithm by changing the grid size at different stages of the SGA:

Fine_0: the entire master-slave SGA works on the fine grid (48 by 24) problem.

Fine_20: the master-slave SGA works on the coarse grid (16 by 8) for the first 20 generations and then changes to the fine grid until convergence.

Fine_40: the coarse grid problem is run for the first 40 generations, and then changed to the fine grid until convergence.

Fine_60: the master slave SGA works on the coarse grid for 60 generations and then changes to the fine grid until convergence.

5 RESULTS AND DISCUSSIONS

Table 4 shows the variations in quality of solutions for the 4 multiscale cases. The quality of solution when the fine grid is used alone is better than all the other cases when part of the SGA works on the coarse grid. In fact, when the SGA spends more time on the coarse grid problem (e.g., for Fine_60), the solution quality deteriorates somewhat. However for this objective function, on direct comparison of solution qualities for the 4 cases, we can also observe that the optimal fitnesses do not vary more than 2%. In all cases the multiscale solution is better than the coarse grid solution alone (Table 2, for k=3).

Also, when the SGA works on the coarse grid problem before shifting to the fine grid, it converges to a solution that indicates that 2 extraction wells should be installed. However, when the problem is solved using the fine grid alone, then the solution indicates that 1 extraction and 1 injection well should be installed. When the SGA starts working on the coarse grid, it moves to a solution sub-space guided by the coarse grid problem. When the fine grid takes over, it only ends up refining solutions within that space and is not able to redirect the SGA towards a different solution sub-space.

Trends for best fitness (figure 7), average fitness (figure 8), log variance (figure 9), and percentage convergence (figure 10) were also traced for the four multiscale cases. Figure 7 shows that there are sudden jumps in fitness values when the grid changes to fine grid, indicating that the higher precision of the fine grid enables substantial solution improvement. Also, Fine_60 takes approximately $1/3^{rd}$ the CPU time it takes to solve the Fine_0 case, with only a 2% worse fitness relative to the best solution (Fine_0 case). This is a good tradeoff to observe between computational effort and solution quality improvement.

Figure 8 shows the trends in average fitness, which start at fitness values of 2×10^5 because initially there are large head and health risk penalties in the fitness evaluations. Once the algorithm starts identifying solutions in the feasible space, the mean fitness increases rapidly to higher values that have negligible or zero penalties. The mean fitness does not have dramatic changes when the grid is changed, compared to the best fitness trend. This shows that the grid changes are introducing improvements in the best members of the population, with less effect on the rest of the population.

Log variance of population fitness (Figure 9) decreases very slowly initially with generations. As the population approaches convergence, the log variance drastically decreases to very low values, as expected. An interesting trend noticed here is that when the SGA starts early with the fine grid problem (e.g., Fine_0, Fine_20) the variance drops steadily. But when the SGA changes to the fine grid problem at a later stage (i.e. Fine_40, Fine_60), a sudden increase in the log variance is observed. When the coarse grid problem approaches convergence, a sudden shift to the fine grid increases the population variance because candidate designs that have approximately the same quality (or fitness value) on the coarse grid are identified as distinctly different designs on the fine grid.

Figure 10, which exhibits the convergence trend for the 4 different cases, indicates that Fine_20, which shifts to the fine grid at the 20^{th} generation, takes the longest time to converge relative to the other 3 cases. Another observation is that even though Fine_0 takes fewer generations to converge than Fine_60, it takes the most CPU time to comverge. Hence, for large scale engineering problems the actual CPU hours invested is a more important issue to consider, than generations.

6 CONCLUSIONS

The above work highlights some of the crucial issues involved in optimizing problems that use information from numerical models with varying spatial grid sizes. As observed in our results, coarse grids and fine grids can drive the optimization search towards different sub-spaces of the solution space because they solve the numerical model with different inaccuracies. These numerical inaccuracies can impart variation in fitness functions evaluations for the same candidate designs. This ends up affecting the optimization search and the time taken for the GA to converge to an optimal or near optimal solution. For example. when the numerical model used only the fine grid it came up with a very different solution, i.e. installation of one injection and one extraction well at the remediation site, as compared to the cases when the GA used the coarse grid fully or partially and converged to a two extraction well solution. In this problem, the strategy of using coarse grid along with the fine grid within a master-slave parallel environment proved to be very useful for saving computational effort with little reduction in solution quality. Such insights on the tradeoffs between solution quality and convergence time are important for numerical modelers who need to understand how sensitive their optimization search is to the design of their numerical model.

Table 4: Solutions for master slave SGA using multiscales at different stages of the GA

Case	Fitness (1000's)	Well 1 location index	Well 1 pumping rate (m^3/day)	Well 2 location index	Well 2 pumping rate (m^3/day)
Fine_0	-180.2	9	-188 (extraction)	34	139 (injection)
Fine_20	-182.2	20	-166 (extraction)	23	-146 (extraction)
Fine_40	-182.4	20	-188 (extraction)	23	-129 (extraction)
Fine_60	-182.9	22	-125 (extraction)	15	-188 (extraction)

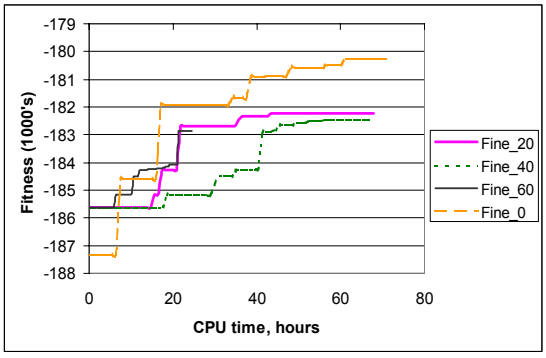

Figure 7. Best Fitness trend

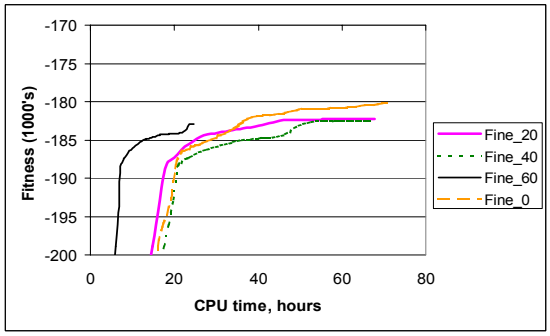

Figure 8. CPU time vs. Average Fitness

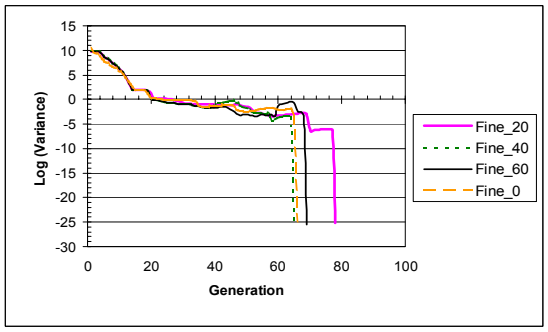

Figure 9. Log variance trend

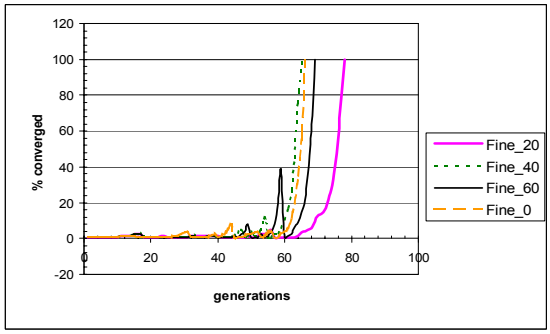

Figure 10. Population convergence trend

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. BES 99-03889, and the U. S. Army Research Office under grant numbers DAAD19-00-1-0389 and DAAD19-001-1-0025.

We would also like to thank Professor David E. Goldberg, Laura A. Albert, and Martin Pelikan for the insightful discussions related to this work.

References

Aksoy, A. And T.B. Culver, Effect of sorption assumptions on aquifer remediation designs, *Groundwater*, 38(2), 200-208, 2000.

Albert L.A., and Goldberg, D.E., Efficient Evaluation Genetic Algorithms under Fitness Functions, IlliGAL Report No. 2001024, July 2001.

Cantu'-Paz, E., A survey of Parallel Genetic Algorithms, *Calculateurs Paralleles, Reseaux et Systems Repartis*, Vol. 10, No. 2, pp. 141-171, Paris: Hermes, 1998.

Cantu'-Paz, E., Designing efficient and accurate parallel genetic algorithms, PhD thesis, 1999.

Clement, T. P. , RT3D - A modular computer code for simulating reactive multi-species transport in 3-Dimensional groundwater aquifers, *Battelle Pacific Northwest National Laboratory Research Report*, PNNL-SA-28967. (http://bioprocesses.pnl.gov/rt3d.htm.), 1997

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

Clement, T. P., Johnson, C. D., Sun, Y., Klecka, G. M., and Bartlett, C., Natural attenuation of chlorinated solvent compounds: Model development and field-scale application, *Journal of Contaminant Hydrology*, 42, 113-140, 2000.

Harik G. R., Cantu-Paz E., Goldberg D. E., and Miller B. L., The gambler's ruin problem, genetic algorithms and the sizing of populations, In *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*, pp 7-12, IEEE press, New York, NY, 1997

Goldberg David E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison–Wesley, New York, NY, 1989

Gopalakrishnan G., Minsker B., and Goldberg D.E., Optimal sampling in a Noisy Genetic Algorithm for Risk-Based Remediation Design, *Journal of Hydroinformatics*, in press, 2002.

Grefenstette J.J. and Fitzpatrick J.M., Genetic search with approximate function evaluations, In Grefenstette, J.J. (Ed.), *Proceedings of an International Conference on Genetic Algorithms and their Applications*, pp 112-120, Hillsdale, NJ, 1985.

Hogg, R., and Craig, A., *Introduction to Mathematical Statistics*. Macmillan Publishing Co., Inc., New York, 1978.

Liu, Y., and B. S. Minsker, "Efficient multiscale methods for optimal in situ bioremediation design." *Journal of Water Resources and Planning Management*, in press, 2001.

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

Reed P., Minsker B. S., and Goldberg D. E., Designing a competent simple genetic algorithm for search and optimization, *Water Resources Research*, 36(12), 3757-3761, 2000

Reed, P. Striking the Balance: Long-Term Groundwater Monitoring Design for Multiple Conflicting Objectives, Ph. D. Thesis, University of Illinois, 2002.

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

Smalley J. B., Minsker B. S., and Goldberg D. E., Risk-based In Situ bioremeditation design using a noisy genetic algorithm, *Water Resources Research*, 36(20), 3043-3052, 2000.

Wang, Q.J., The genetic algorithm and its application to calibrating conceptual runoff models, *Water Resource Research*, 27(9), 2467-2471, 1991.

Wang, M. And C. Zheng, Optimal remediation policy selection under general conditions, *Groundwater*, 35(5), 757-764, 1997.