MULTISCALE PARALLEL GENETIC ALGORITHMS FOR OPTIMAL GROUNDWATER REMEDIATION DESIGN

BY

MEGHNA BABBAR

B.E., University of Roorkee, 2000

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Environmental Engineering in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2002

Urbana, Illinois

ABSTRACT

Genetic algorithms (GAs) have been shown to be powerful tools for solving a wide variety of water resources optimization problems. Applying these approaches to complex, large-scale applications, which is usually where these methods are most needed, can be difficult due to computational limitations. In groundwater remediation design problems, computational limits are often driven by the spatial grids required to simulate the performance of candidate designs. Fine grids usually improve the accuracy of the solutions, but they also pose major bottlenecks in the computational efficiency of the algorithms. In this work, we present multiscale parallel genetic algorithms that can be used to improve the performance of water resources management problems that have spatial grids. Two management techniques, that use the advantages of parallel computing, are employed to test the significance of spatial grid sizes on optimization search. The first technique uses a simple master-slave parallel setup to change to different grid sizes at different stages of the simple genetic algorithm (SGA), whereas the second technique uses an island injection parallel SGA (IIGA) to investigate the performance effects when the optimization algorithm has different populations working on different spatial grids on different processors, allowing frequent communication between the processors. Results for the two techniques highlight two important observations. First, objective functions that use numerical models can drive the GAs towards different sub spaces of the optimum solution space, depending upon the spatial grid size. Second, tradeoff between solution quality improvement and computational time invested by using such techniques is crucial for numerical modelers, who use such optimization algorithms, to understand. The master-slave multiscale approach reduced computational time by 67% with only a 2%

increase in the optimal cost. The IIGA proved to be an inefficient approach for this problem as it is currently configured, despite success with this approach in other fields. Further research is needed to explore whether other IIGA configurations would be more efficient.

To my parents and Sunny uncle

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to my thesis advisor Barbara S. Minsker who has been an endless source of information and guidance. A few sentences are not enough to convey how much she has inspired me in the past two years. It has been an honor for me to have her as a mentor and as a friend.

I would like to thank the faculty members of the Civil & Environmental Engineering Department at University of Illinois at Urbana-Champaign, faculty members of the Pulp & Paper Technology Department at University of Roorkee (India), and all my school teachers for giving me an invaluable education. Each person contributed a piece of the puzzle to complete the big picture so that it all made sense.

I would like to thank all my research group members: Abhishek Singh, Felipe Espinoza, Patrick Reed, Rachel Arst, Shengquan Yan, William Michael, and Xiaolin Ren, for the wonderful support system they created for everybody in the group, with their extraordinary talents and cheerful spirits. Their presence was always reassuring, especially when times were tough.

I'd also like to thank Professor David E. Goldberg, Laura Albert, and Martin Pelikan from the General Engineering Department, University of Illinois at Urbana-Champaign; and Dr Bruce Loftis, and Prasanna Parthasarthy from the National Center for Supercomputing Applications, for the insightful discussions and technical support related to this work.

I would also like to acknowledge the agencies: 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 for funding this research.

Last, but not the least, I’d like to thank my entire family and all my friends for their invaluable support and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Use of optimization techniques along with simulation models to identify cost effective designs for groundwater containment and remediation has gained considerable popularity in the recent past. For example, *Chan* (1994), *Culver and Shoemaker* (1997), *Karatzas and Pinder* (1996), *Lee and Kitanidis* (1991), *McKinney and Lin* (1996), *Minsker and Shoemaker* (1998), *Misirli and Yazicigil* (1997), *Ritzel et al* (1994), *Rizzo and Dougherty* (1996), *Sawyer et al* (1995), *Sun et al* (1996), *Tiedeman and Gorelick* (1993), and *Zhen and Uber* (1996) have implemented various optimization methods for groundwater management, including nonlinear and linear programming, optimal control, genetic algorithms, simulated annealing and cutting plane methods.

In these optimization models, a numerical simulation model is used to analyze the performance of candidate remediation designs. The degree of accuracy of these numerical fate and transport models used in groundwater management 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 thesis, we explore the potential for two new multiscale parallel approaches to improve the performance of an SGA-based groundwater remediation design optimization problem. The new approaches investigated are:

a) Multiscale single population parallel genetic algorithm with a simple master-slave configuration.
b) Multiscale multi-population island injection genetic algorithm.

The organization of the thesis is as follows. We first begin with an introduction to the groundwater remediation problem and the design case study, in chapter 2. Chapter 3 discusses the design of an SGA and its parameter settings for this problem. The design problems with different spatial grids are solved individually using the SGA. Chapter 4 presents the multiscale master-slave parallel GA and tests its performance on the case study. Chapter 5 presents the design of a multiscale multi-population island injection genetic algorithm, including the results for the case study. Finally, the results for all of the experiments and approaches are discussed and concluded in chapter 6.

2. GROUNDWATER REMEDIATION DESIGN CASE STUDY

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 contaminated groundwater for each candidate remediation plan and the optimization model estimates the cost (in dollars) and the associated health risks. Below is a description of the two models.

2.1 Numerical model

The groundwater management model is 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 assumes steady state flow of groundwater in the aquifer. The hydraulic conductivity (average 2255.7 m/year) data were generated for a fine grid of 144 by 72 elements using the conditional simulation technique described by *Smalley et al* (2000). 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 and the direction of flow from left to right in Figures 1 and 2. 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. 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. Although biodegradation reactions are often present with BTEX, they are assumed absent in this work to allow rapid solution of the different solution approaches. Hence, the 'no reaction/tracer transport' chemical reaction package in RT3D was used for modeling contaminant transport. 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 downgradient of the boundary on the right side of Figure 3. See *Smalley et al* (2000) for more details on the risk assessment model.

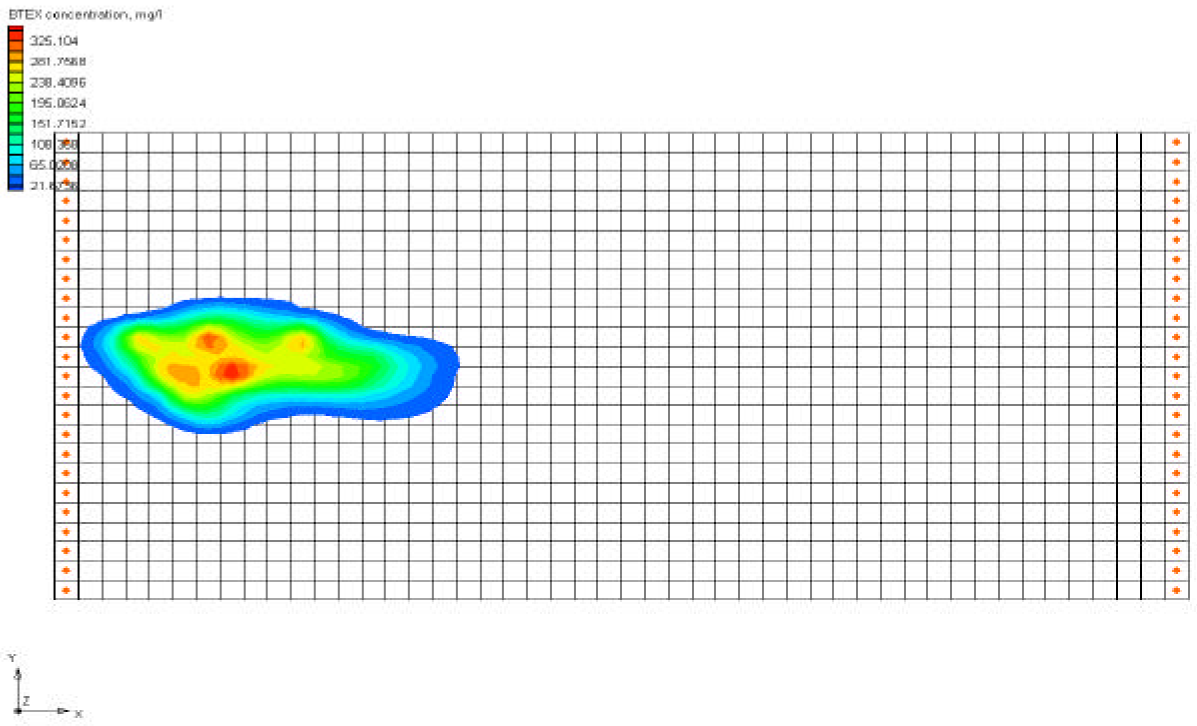


Figure 1. BTEX plume on '48 by 24' grid size'

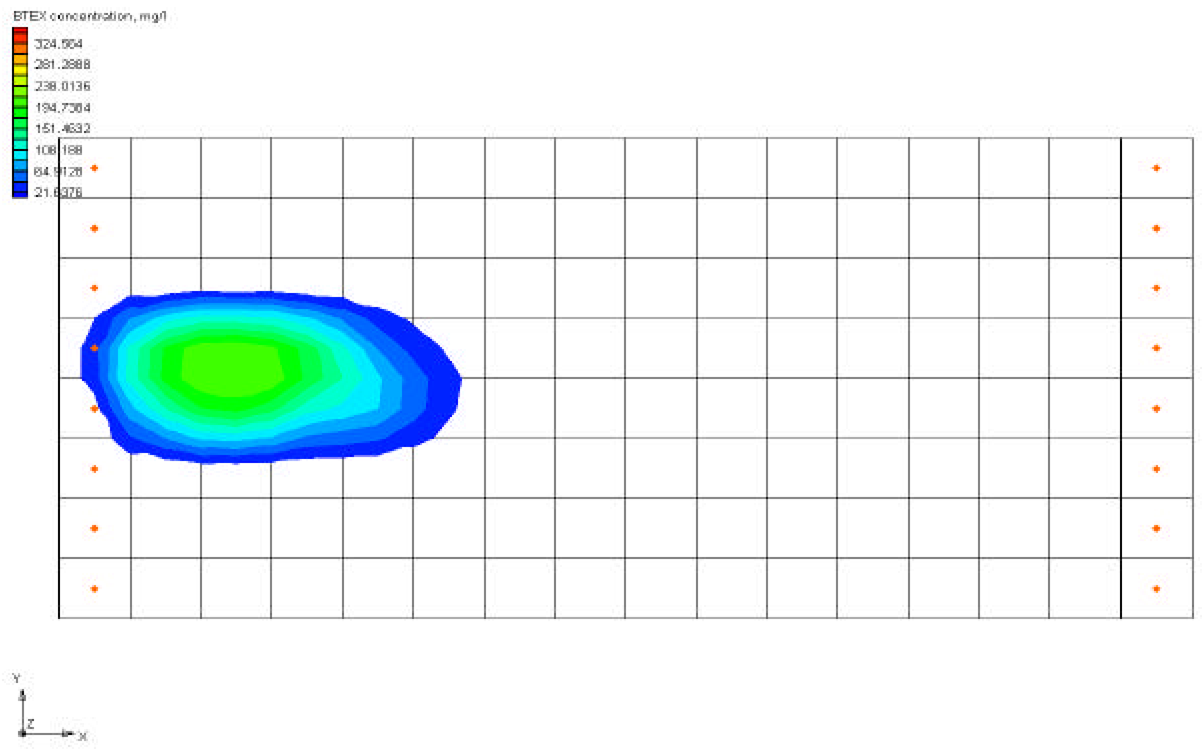


Figure 2. BTEX plume on '16 by 8' grid size'

2.2 Optimization model

The objective of the remediation 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]:

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

The constraints for the problem are:

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

$$Risk_{t,k}^{TOTAL} = Risk_{t,k}^{w} + Risk_{t,k}^{shw} + Risk_{t,k}^{nc} \leq TR \; \forall t, \forall k \tag{2}$$

where,

$Risk_{t,k}^{w}$, is cancer risk due to ingestion of contaminated drinking water,

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

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

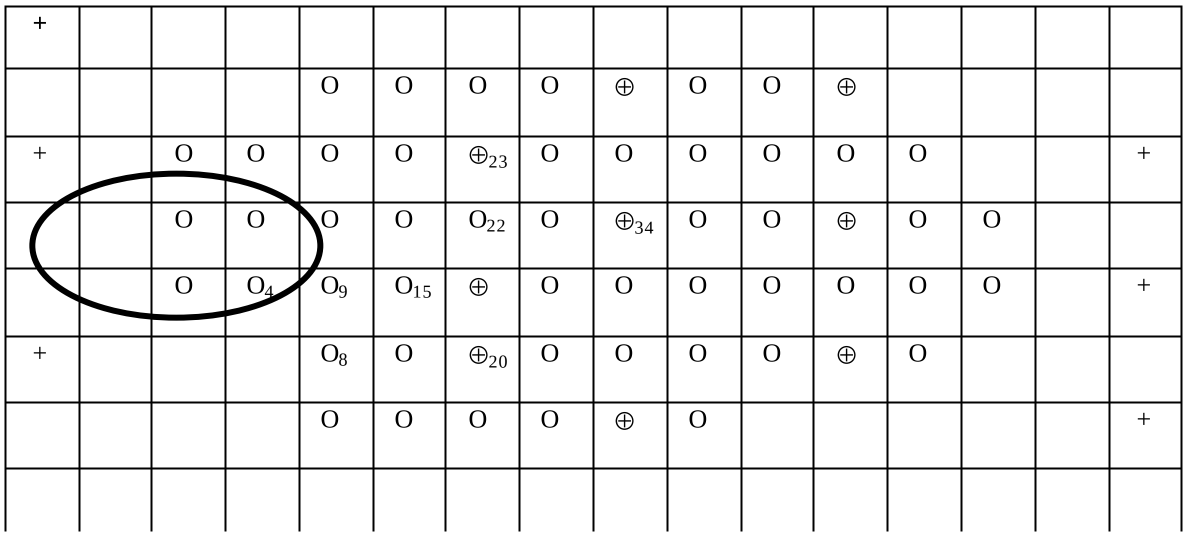
See *Smalley et al.* (2000) for details on how the risk is calculated.

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*.

$$u_{\min,i} \leq |u_i| \leq u_{\max,i} \quad \forall i \tag{3}$$

3) 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*.

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



+	Monitoring wells (15)
O	Possible remediation well locations (58)
⬭	Plume

Figure 3. Plan view of the case study aquifer (grid size 16 by 8)

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. 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. Unlike the usual GA operations where fitness functions are maximized, the fitness function here is minimized:

$$\textit{Fitness function} = C_{TOT} + w_1 * \textit{Risk violation} + w_2 * \textit{head violation} \qquad (5)$$

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 ALGORITHMS

Simple genetic algorithms (SGAs) have been shown to be good optimization tools for various water resources problems [*Aly and Peralta, 1999, Cieniawski et al., Dandy and Engelhardt, 2001, 1995, Ritzel et al., 1994, Smalley et al., 2000, Wang, 1991, Wang and Zheng, 1997, Gopalakrishnan et al., 2001*]. Their main advantage over conventional gradient based methods lies in their ability to solve discrete, non-convex, and discontinuous problems [*Goldberg, 1989*]. SGAs emulate natural selection and genetics to produce optimal designs. They work with ‘strings’ of decision variables mapped in binary space (also known as ‘chromosomes’), and search from a population of possible designs (“individuals”) using the information provided by the objective function (“fitness function”). Using three probabilistic operators - reproduction, crossover, and mutation - the SGA evolves the population to better solutions until it converges to the optimal or near-optimal solution.

3.1 Design methodology

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 GA design methodologies to identify realistic parameter values. A brief overview of their suggestions that we have used in our work is given below.

3.1.1 Population size

GA theory shows 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:

$$N \geq -2^{k-1} \ln(a) \frac{s_{bb} \sqrt{p\ (m-1)}}{d} \tag{6}$$

where,

N = population size,

k = BB order,

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

d = signal difference,

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

In this equation, the term '$s_{bb}\sqrt{p\ (m-1)}$' is approximated as the standard deviation of fitness s_f for a large initial population (*Reed et al,* 2001). 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 's_f' was estimated to be 1,472,050 for the '16 by 8 grid' problem and 1,730,517 for the '48 by 24 grid' problem.

The signal difference 'd', which represents the difference in fitness between best and second best members of the population, is evaluated 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 evaluated above and organizes them into classes that are probabilistically distinct. Based on this frequency analysis theory, the number of bins created are approximately equal to $\sqrt{n}$, where n is the total number of individuals (1000 in this case). The number of individuals per bin are evaluated based on the condition that np_i should be ≥ 5, where p_i is the percentage of the total number of individuals that are 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.

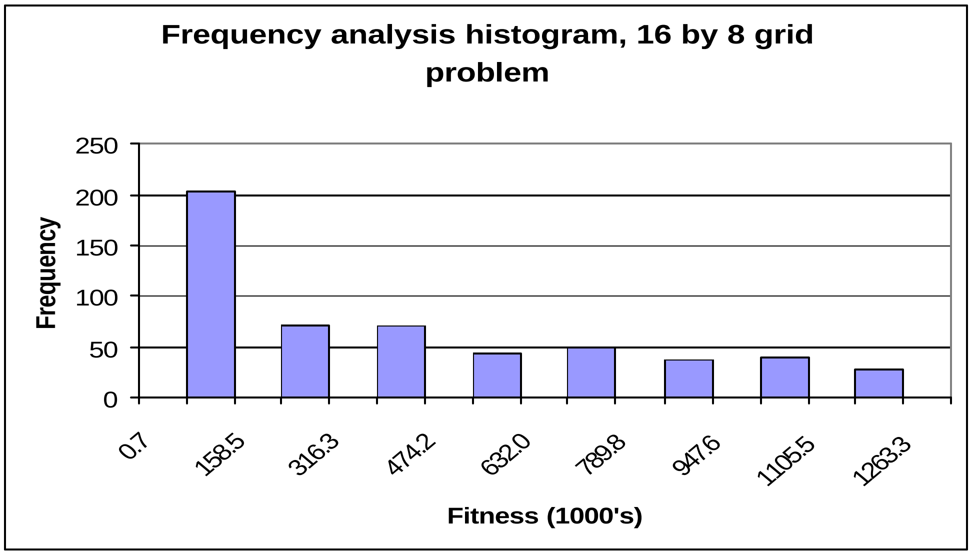


Figure 4. Histogram for initial population, 16 by 8 grid problem

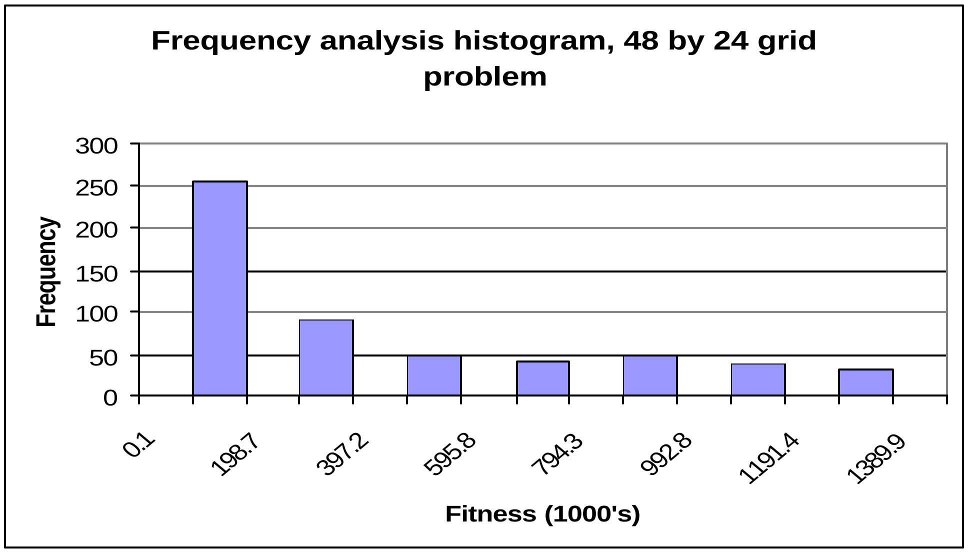


Figure 5. Histogram for initial population, 48 by 24 grid problem

Since we cannot identify the actual building block order 'k' in equation 6, 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

3.1.2 GA operators

A tournament size of 's' equal to 4 was used for this problem. Tournament size is the number of individuals that compete in the selection process. Per Reed et al. (2000), probability of crossover (P_c) was estimated to be approximately '(s-1)/s' or 0.75 in this case. P_c is the parameter that exerts the innovative force on the fit individuals to create new diverse designs. Probability of mutation (P_m), which assists in local refinement of solutions, is estimated to be equal to 1/N, where N is population size (*De Jong et al*., 1975). For a tournament size of 4, convergence time for SGA is estimated to be less than twice the chromosome length (*Thierens et al*., 1998), so the maximum number of generations was set to 96 for the string length of 48.

The $m + l$ selection scheme, without replacement, was used for selecting the best individuals for successive generations, where m represents the child population and l 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 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 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.

A comparison of 'without injection' and 'with injection' techniques for solving the SGA (Figure 6) for k= 1 to 4 illustrates the advantage of the injection method for this problem. The solution quality of the SGA with injection is much better than that for SGA without injection, and the computational time is also faster. For this reason, the injection method was used for all subsequent cases in this thesis.

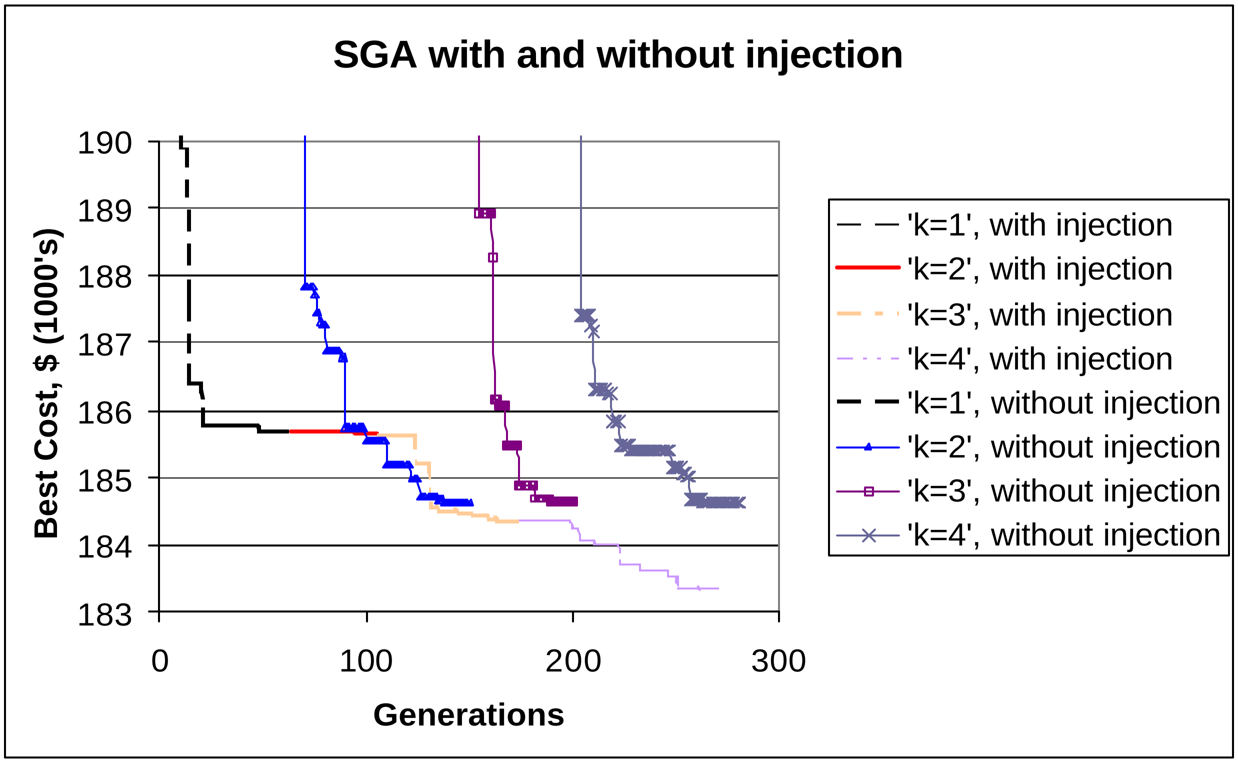


Figure 6. Comparison of SGA with and without injection technique (grid size 16 by 8)

3.2 SGA results

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.

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

Building block order, k	Population size	Best fitness, ($*0.001)	Convergence time (# of generations)	Percentage of population converged
1	30	185.7	62	100%
2	60	185.6	42	100%
3	120	184.4	67	100%
4	240	183.3	96 (maximum allowed)	37.5%

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

Building block order, k	Population size	Best fitness, ($*0.001)	Convergence time (# of generations)	Percentage of population converged
1	25	183.9	96	4%
2	50	183.9	45	100%
3	100	182.1	58	94%
4	200	180.6	51	100%

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 less than that for 'k=4', but the solution quality varies less than 2% from the best solution identified in each case.

An interesting observation is that, for ‘k=3’, the optimal solution for ’16 by 8 grid’ problem is actually equivalent to a fitness of $186.06*10^3 on the ’48 by 24 grid’ size as compared to $184.4*10^3 (Table 2) on the ’16 by 8 grid’ size. Hence, the optimal cost for ‘k=3’ is more expensive when the problem is solved using the coarse grid size than when the SGA is solved using fine grid problem (Table 3). The cost error on the coarse grid is due to errors in the hydraulic heads, which are used to estimate the electricity costs in the cost function (see Smalley et al, 2000, for details).

4. MULTISCALE MASTER-SLAVE PARALLEL GENETIC ALGORITHMS

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 7). Master-slave GAs search the decision space in the same way as an SGA, but can be solved more quickly.

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.

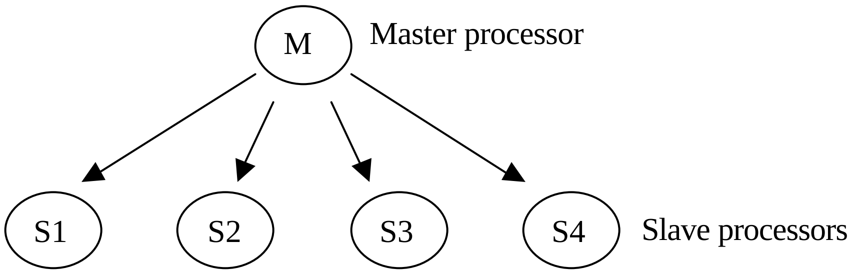


Figure 7. Master and slave parallel processors

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.

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:

1. Fine_0: the entire master-slave SGA works on the fine grid (48 by 24) problem.
2. 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.
3. Fine_40: the coarse grid problem is run for the first 40 generations, and then changed to the fine grid until convergence.
4. Fine_60: the master slave SGA works on the coarse grid for 60 generations and then changes to the fine grid until convergence.

4.1 *Results and discussion*

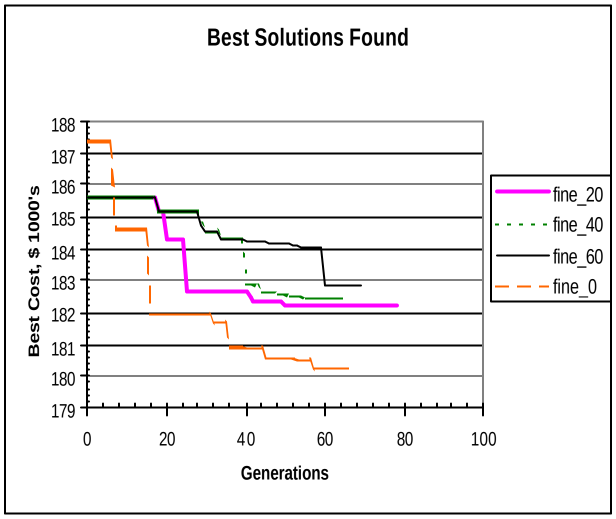
Table 4 shows the variations in quality of solutions for the 4 multiscale cases. We can observe that the coarse and fine grid problems do produce dissimilar results, as observed previously with the SGA. The quality of solution when the fine grid is used alone is better than all the other cases when the SGA works partially 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. In all cases the multiscale solution is better than the coarse grid solution alone however (see Table 2).

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

Case	Fitness or cost*0.001, $	Well 1 location index	Well 1 pumping rate (m^3/day)	Well 2 location index	Well 2 pumping rate (m^3/day)	Well 3 location index	Well 3 pumping rate (m^3/day)
Fine_0	180.2	9	-188 (extraction)	34	139 (injection)	Not installed	0.0
Fine_20	182.2	20	-166 (extraction)	23	-146 (extraction)	Not installed	0.0
Fine_40	182.4	20	-188 (extraction)	23	-129 (extraction)	Not installed	0.0
Fine_60	182.9	22	-125 (extraction)	15	-188 (extraction)	Not installed	0.0

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 is moved to a solution space guided by the coarse grid problem, which indicates installation of 2 extraction wells. When the fine grid takes over, it ends up refining solutions in that space only and is not able to redirect the SGA towards a different solution space. Direct comparison of solution qualities for the 4 cases indicates that the optimal costs do not vary more than 2% however.

Trends for best cost (figure 8), mean cost (figure 9), log variance (figure 10), and percentage convergence (figure 11) were also traced for the four SGA cases for the 'k=3' run. Figure 8 shows that the fitness evaluated for the same best individual (obtained from 'k=2' problem through 'injection') at generation 0 is larger on the fine grid (Fine_0) than on the coarse grid (Fine_20, Fine_40, Fine_60), since only Fine_0 starts on the fine grid immediately at 'k=3'. For Fine_20, Fine_40, and Fine_60 cases there are sudden drops in fitness values when the grid changes to fine grid, indicating that the higher precision of the fine grid enables substantial solution improvement. Figure 9 shows the trends in mean costs, which start at fitness values of 2 x 10^5 because initially the mean costs are very high due to large penalty values in the fitness evaluations. Once the algorithm starts identifying solutions in the feasible space, the mean costs decrease rapidly to lower values that have negligible or zero penalties. The mean costs do not have dramatic changes when the grid is changed as the best costs did. This shows that the grid changes are introducing improvements in the best members of the population, with slower effects on the rest of the population. Log variance of population fitness (Figure 10) 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. At later generations the coarse grid problem starts approaching convergence, but when the grid is shifted to the fine grid, the population variance increases. This effect likely occurs 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 11, which exhibits the convergence trend for the 4 different cases, indicates that Fine_20, which shifts to the fine grid at the 20th generation, takes the longest time to converge relative to the other 3 cases.



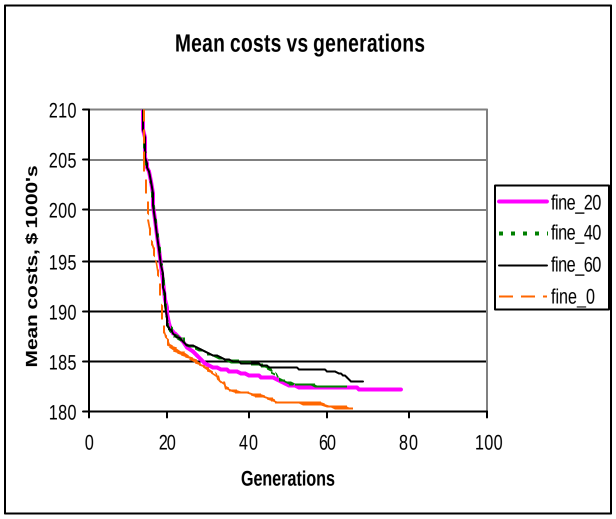
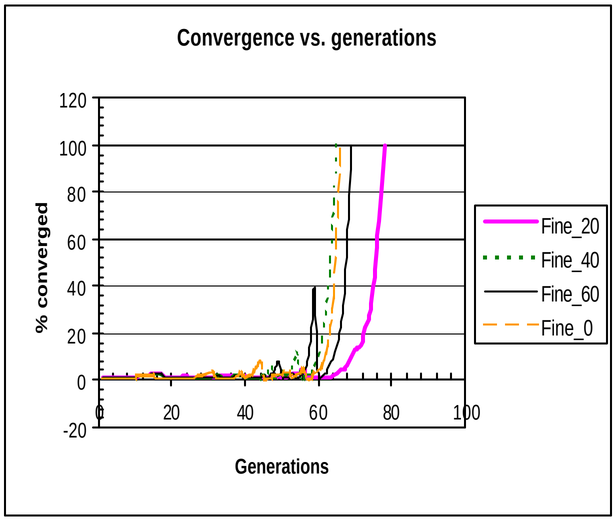


Figure 8. Best cost trend

Figure 9. Mean cost trend



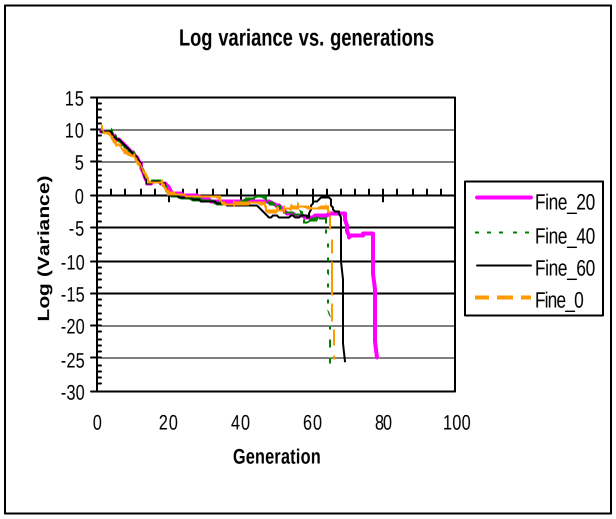
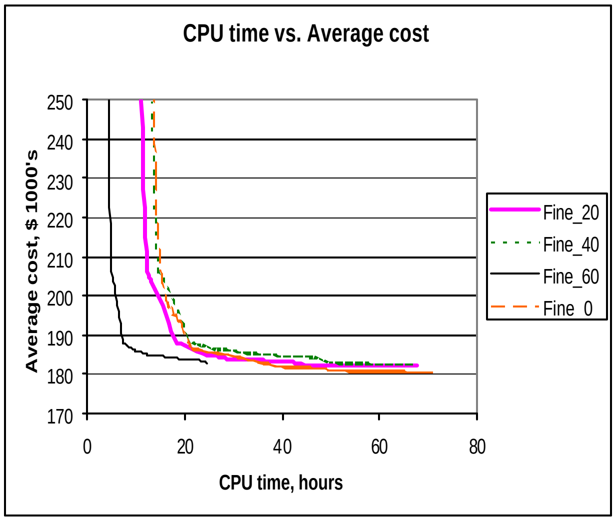


Figure 10. Log variance trend

Figure 11. Population convergence trend

Figure 12 and Figure 13 show trends in best costs and average costs with respect to CPU time. CPU time includes actual execution time, waiting time for processors, input/output time, and other overhead time that is spent in successfully completing the job. Both figures show that Fine_60 takes approximately 1/3rd the CPU time it takes to solve the Fine_0 case, with only a 2% increase in cost relative to the best solution (Fine_0 case). This is a good tradeoff to observe between computational effort and solution quality improvement.



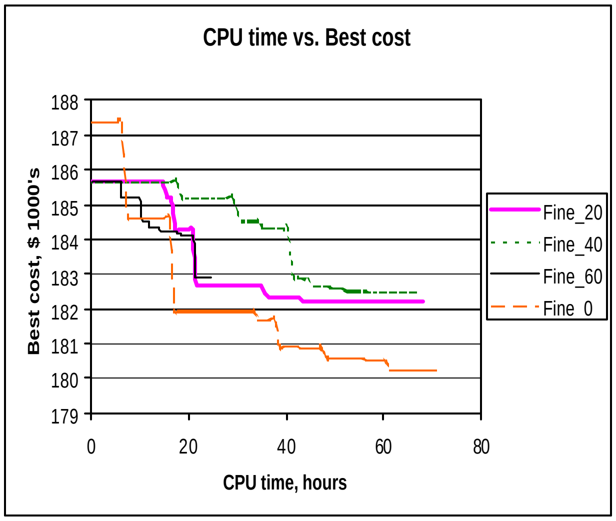


Figure 12. CPU time vs. best cost

Figure 13. CPU time vs. average cost

5. MULTISCALE ISLAND INJECTION GENETIC ALGORITHMS

Island injection genetic algorithms are multi-population parallel GAs that are also called 'Distributed' GAs. Unlike a simple master slave parallel GA, they consist of groups of master-slave GAs ("islands") that work with separate populations and occasionally migrate individuals between each island. Each master-slave island searches its own space for feasible solutions. The islands can be interconnected through various topologies: Bidirectional rings, unidirectional rings, hypercubes, tree structure, etc. [*Cantu'-Paz* ,1999; *Malott et al*, 1996; *Eby et al*, 1999]. Different strategies for deciding migration rates and frequencies can be adopted. *Tanese et al* (1987), *Whitley et al* (1990), *Eby et al* (1999) and *Punch et al* (1995) have experimented with different fixed migration frequencies between different islands, whereas *Cantu'-Paz* (1999) migrated individuals only after a certain convergence criterion for the entire deme was met.

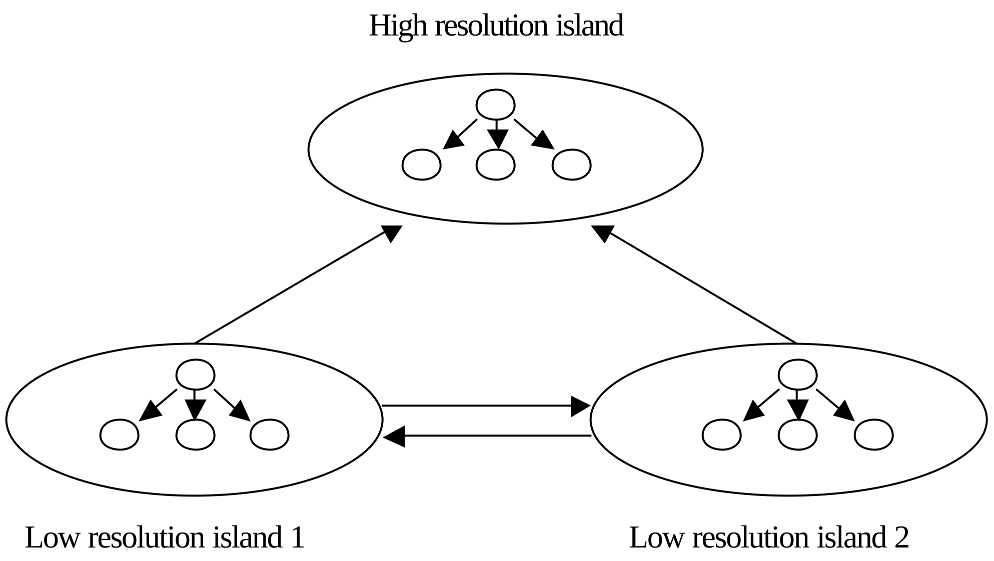


Figure 14. Island injection topology

Similar to the work done by *Malott et al* (1996) and *Eby et al* (1999), the topology of the IIGA used in this study consists of three islands that search for solutions within their own space for the same problem, but at different resolutions. Two islands work on the problem at lower resolution (in our case, a coarse grid) and they periodically inject solutions (for fine tuning) to the island working at higher resolution. The two 'coarse' islands also exchange solutions between each other at periodic intervals to push each other towards the global optima. Since the coarse islands work on a more 'inexact' representation of the numerical grid, they take much less time to compute the solutions. The good 'inexact' solutions passed into the 'finer' island provide initial solutions that the high resolution island can then "polish" to a more accurate representation of the solution. See Figure 14 for the architecture.

Cantu'-Paz (1999) found that as migration rates increased, the probability of the population converging sooner also increased. He also observed that above a 10% migration rate there was not much improvement in the convergence. In this study, three different migration rate schemes have been used for comparison purposes:

a) Migration rates between all islands are equal to 10%
b) High migration rate (10%) between low resolution islands and low migration rate (5%) from low islands to high resolution island.
c) Low migration rate (5%) between low resolution islands and high migration rate (10%) from low islands to high resolution island.

Migration between all islands takes place every 10 generations by migrating the best individuals from one island to the other and replacing the worst individuals there. Though transferring of best individuals should result in exchanging of good building blocks, excessive migration could lead to premature convergence. This possible effect will also be observed during the experiments.

The GA parameters used for all three islands are identical. They all work with the entire chromosome and search the entire decision space. The only difference between them lies in the resolution of the fitness function, population sizes and random number seeds. Since the jobs are performed on the Origin 2000’s dedicated queues, the number of optimal processors used for each island is 1/3rd the total number of available processors on the dedicated queue.

Initial experiments are performed using population sizes for all islands equal to 1 and 2/3rd times the population size calculated for the SGA, respectively, for all three migration rate schemes. Once the optimal migration scheme for the islands is identified, the best migration scheme is implemented on a smaller problem using 1/3rd the SGA population size for all islands. A problem with irregular population size, where the population of the coarse islands is twice that of the fine island, was also tested to better balance processor loads, since the coarse grid problem takes about half the CPU time of the fine grid. Tradeoffs between solution quality and computational effort were assessed for each of these experiments. Table 5 summarizes these different test runs.

Table 5. Summary of IIGA test runs

Test case	Population size	Coarse to fine migration rate	Coarse to coarse migration rate
Pop124_10f_10c	124 each island	10%	10%
Pop124_10f_5c	124 each island	10%	5%
Pop124_5f_10c	124 each island	5%	10%
Pop84_10f_10c	84 each island	10%	10%
Pop84_10f_5c	84 each island	10%	5%
Pop84_5f_10c	84 each island	5%	10%
Pop40_10f_10c	40 each island	10%	10%
Pop40f-84c_5f_5c	40 for fine island, 84 for coarse islands	5%	5%

5.1 *Results and discussion*

Figures 15 and 16 depict trends for best costs and average costs with respect to CPU time for all of the fine islands in the experiments, respectively. A comparison of all of the fine islands suggests that a higher migration of individuals to the fine islands results in better solution qualities. For example, in figure 15, solution quality for cases when population size is 124 or 84 for all 3 islands is better when the migration rate is 10% than 5% from coarse islands to fine island (compare cases pop124_10f_10c and pop124_10f_5c with case pop124_5f_10c, e.g.). Also, when the migration rate is higher (i.e. 10%) from coarse islands to the fine island, the convergence is much faster (compare cases pop124_10f_5c and pop124_10f_10c, e.g.). The migration scheme with 10% migration rate between all islands was applied to a smaller population size for the 3 islands, i.e. 40 each ('pop40_10f_10c'). This experiment took approximately 5% less cpu time than the base Fine_0 case, but had results that were no more than 2% worse as compared to Fine_0. The irregular population case 'pop40f-84c_5f_5c' did not behave similarly to 'pop40_10f_10c' case either in terms of solution quality or cpu time.

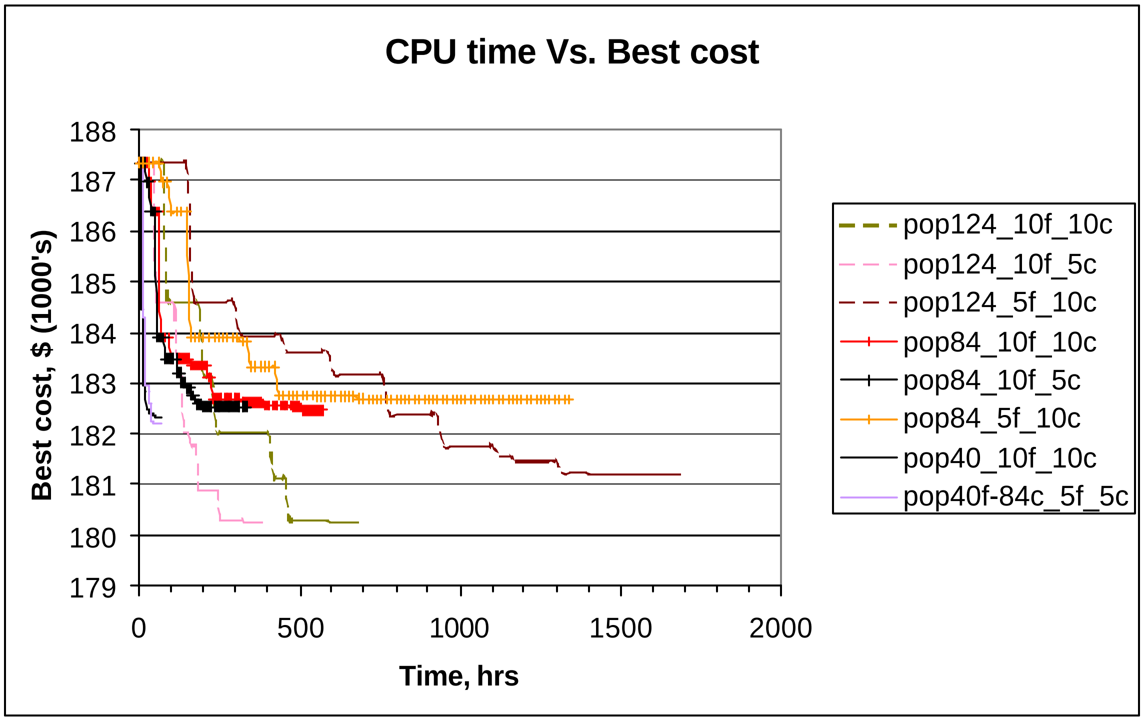


Figure 15. Trends for best costs vs. CPU time for all the fine islands

In figure 16 we can observe that the best cost in the smallest population size case (pop40_10f_10c), and the irregular case (pop40f_84c_5f_5c) took over the population much sooner with respect to cpu time as compared to the other cases which had larger population sizes. However, the results for these 2 cases were only 2% more expensive as compared to other cases that had larger population sizes and took more time to converge. Hence using these 2 faster cases seems to be a better proposition for this objective function, since with very little degradation in solution quality they have saved a large amount of computational time relative to the other IIGA configurations ('pop40_10f_10c' is 96% faster than 'pop124_5f_10c').

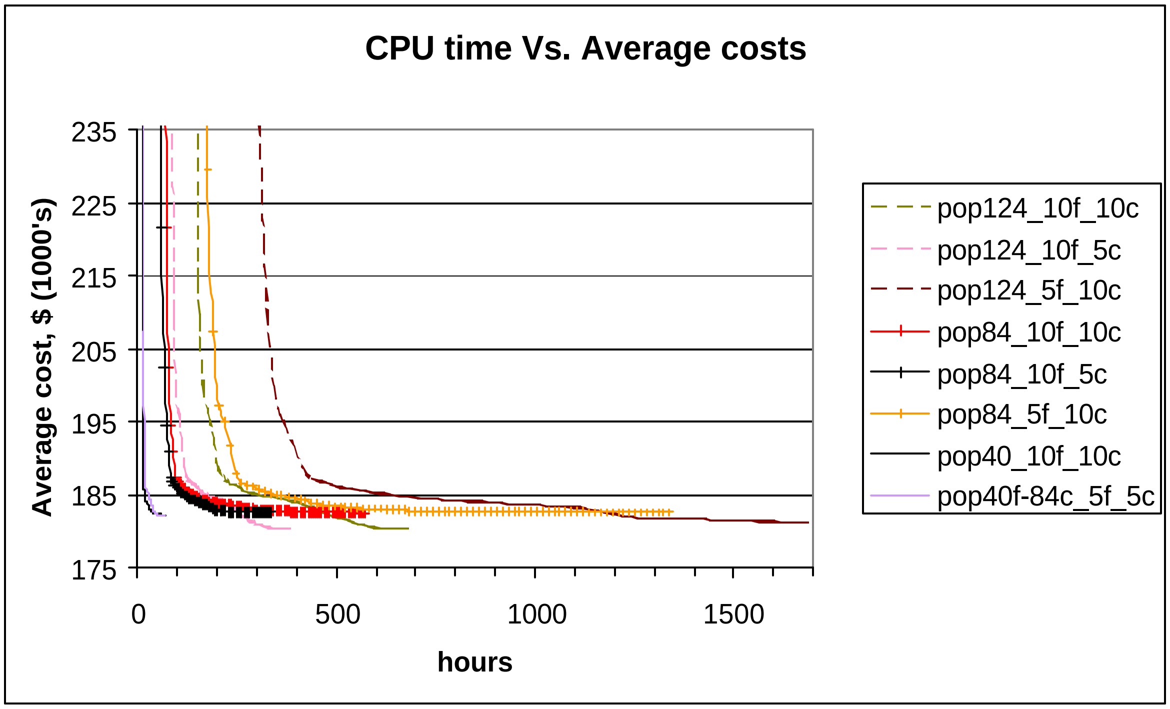


Figure 16. Trends for average costs vs. CPU time for all the fine islands

Figures 17 and 18 show comparisons of the best cases for the multiscale master-slave GA and the multiscale IIGA for our optimization problem. These graphs indicate that Fine_0 and pop124_10f_5c, which use population sizes of 124, have the best solution quality compared to all other experiments. However, the computational expense for attaining this level of quality is considerably higher for the multiscale IIGA. Fine_0 uses 82% less time to compute this solution than pop124_10f_5c. On the other hand, Fine_60 takes the least CPU time to solve the problem (65% less than Fine_0), but also converges to the worst solution quality (1.45% worse than Fine_0). Pop40_10f_10c, which uses the minimum population size for the islands, converges to a solution quality 0.3% better than Fine_60 but 1.14% worse than Fine_0. Also, pop40_10f_10c takes approximately the same time as Fine_0 to arrive to this inferior solution. These results illustrate the tradeoffs between solution quality and computational time. It is clear, however, that the IIGA in its current configuration is worse than the multiscale master-slave approach or solving the entire problem on a fine grid.

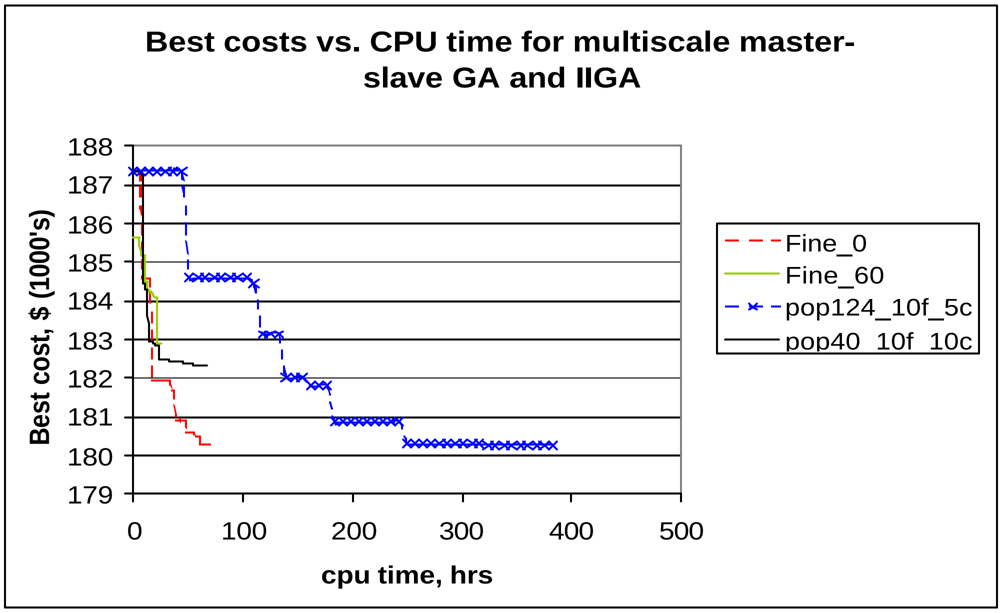


Figure 17. Comparison of best costs for best cases in multiscale master-slave GA and IIGA

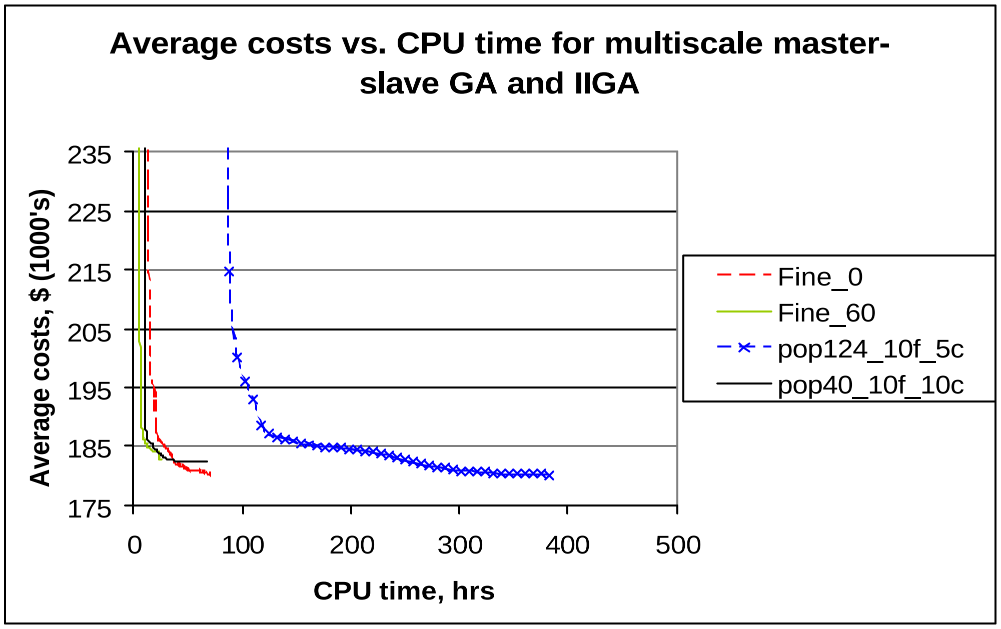


Figure 18. Comparison of average costs for best cases in multiscale master-slave GA and IIGA

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 accuracies. These numerical inaccuracies can impart variations in fitness function evaluations for the same candidate designs. This ends up affecting both the optimal solutions found and the time taken for the GA to converge to an optimal or near optimal solution. For example, in the multiscale master-slave GA, 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 partially or fully and converged to a two extraction well solution.

For this problem, the strategy of using the coarse grid along with the fine grid within a master-slave parallel environment proved to be very useful for saving computational effort (up to 67% savings for Fine_60 with respect to Fine_0) with little reduction in solution quality (less than 2% for all cases with respect to Fine_0). 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.

The island injection scheme was not as computationally robust as the multiscale master-slave GA for our problem. It took 82% more time for the IIGA to arrive at the same solution as the fine grid case (Fine_0), with no improvement in solution quality. This result is surprising because the IIGA has been highly successful at solving problems in other fields (*Eby et al,* 1997; *Eby et al,* 1999; *Malott et al,* 1996; *Punch et al,* 1995). One possible explanation is that our parallel implementation is inefficient. Speedups ranged from 25 to 85, far less than the theoretical goal of 123 (for 123 processors), so the computing times given here may include substantial processor waiting time. Further research is needed to identify more efficient parallel implementations. Further research is also needed to investigate whether other IIGA configurations would be more efficient, for example using more low-level populations. It is also possible that the results are an artifact of the particular case study used in this work and that it might perform better on more difficult problems. The fitness function used for this case study is very flat and not very multi-modal (i.e., many designs have similar fitness values). This characteristic means that the information passed from the coarse grid populations may not be as valuable as it would be for a more complex function. However, all of the coarse islands in the IIGA had much better solution quality than the SGA working on the coarse grid alone (Table 2, for 'k=3'), which shows that the multi-population approach has some benefits. The IIGA technique may be more promising for more complex fitness functions that are more multi-modal in nature, which will be the basis of future investigation.

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.

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

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

Chan, N., Partial infeasibility method of chance constrained aquifer management, *Journal of Water Resources Planning and Management*, 120(1), 70-89, 1994.

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.

Culver, T.B., and C.A. Shoemaker, Dynamic optimal ground-water reclamation with treatment capital costs, *Journal of Water Resources Planning and Management*, 123(1), 23-29, 1997.

Eby D., Averill R. C., Gelfand B., Punch W. F., Mathews O., Goodman E. D., An Injection Island GA for Flywheel Design Optimization, *Invited Paper, Proc. EUFIT '97, - 5th European Congress on Intelligent Techniques and Soft Computing***,** Sept., 97, 1997

Eby D., Averill R., Goodman E., and Punch W., The Optimization of Flywheels Using an Injection Island Genetic Algorithm, in Bentley, P. (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann, San Francisco, 1999, pp.167-190.

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.

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

Lee, S.I., and P.K. Kitanidis, Optimal estimation and scheduling in aquifer remediation with incomplete information, *Water Resources Research*, 27(9), 2203-2217, 1991.

Lin S-C., Goodman E.D., Punch W.F., Investigating Parallel Genetic Algorithms on Job Shop Scheduling Problems, *Evolutionar Programming VI, Proc. Sixth Internat. Conf., EP97, Springer Verlag, NY, P. J. Angeline, et al., eds.,Indianapolis*, pp.383-394, June, 97, 1997

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.

Malott B., Averill R.C., Goodman E.D., Ding Y., Punch W.F., Use of Genetic Algorithms for optimal design of laminated composite sandwich panels with bending-twisting coupling, presented at *AIAA SDM (Structures, Dynamics and Materials)*, Apr 96, 1996

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.

McKinney, D.C., and M.-D. Lin, Pump-and-treat ground-water remediation system optimization, *Journal of Water Resources Planning and Management*, 122(2), 128-136, 1996.

Miller B. L., and Goldberg D. E., Noise, Sampling and Efficient Genetic Algorithms, IlliGAL Report No. 97001, May 1997

Minsker, B.S., and C.A. Shoemaker, Dynamic optimal control of in situ bioremediation of groundwater, *Journal of Water Resources Planning and Management*, 124(3), 149-161, 1998.

Misirli, F., and H. Yazicigil, Optimal ground-water pollution plume containment with fixed charges, *Journal of Water Resources Planning and Management*, 123(1), 2-14, 1997.

Punch W.F., Averill R.C., Goodman E.D., Lin S-C., Ding Y., Design using Genetic Algorithms – Some results for laminated composite structures., IEEE expert, vol 10 (1), pg 42-49, February 95, 1995.

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.

Rizzo, D.M. and D.E. Dougherty, Design optimization for multiple management period groundwater remediation, *Water Resources Research*, 32(8), 2549-2561, 1996.

Sawyer, C.S., D.P. Ahlfeld, and A.J. King, Groundwater remediation design using a three-dimensional simulation model and mixed integer programming, *Water Resources Research*, 31(5), 1373-1385, 1995.

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.

Sun, Y.-H., M.W. Davert, and W.W.-G. Yeh, Soil vapor extraction system design by combinatorial optimization, *Water Resources Research*, 32(6), 1863-1873, 1996.

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.

Whitley D., Starkweather T., Genitor II : A distributed genetic algorithm, *Journal of Experimental and Theoretical Artificial Intelligence*, 1990.

Tanese R., Parallel genetic algorithm for a hypercube, Grefenstette J.J., Ed., *Proceedings of the Second International Conference on Genetic algorithms*, 434-439. Morgan Kaufmann

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

Zhen, C., and J.G. Uber, Reliability of remediation designs in presence of modeling error, *Journal of Water Resources Planning and Management*, 122(4), 253-261, 1996.

APPENDIX A. RESULTS FOR MULTISCALE ISLAND INJECTION GENETIC ALGORITHM CASES

The following are the descriptions of the various configurations that were tested for the multiscale island injection genetic algorithm and more detailed results.

a) ***pop124_10f_10c***: This experiment used a population size of 124 for each of the coarse islands and the fine island. Migration rate was 10% from coarse islands to the fine island, and 10% between the coarse islands. The job was submitted on NCSA's Origin 2000 dedicated queue, and 41 processors per island were used. The job ran on the Forseti1 machine, and took approximately 26.33 hours of run time and 682.416 hours of CPU time. Table 6 and Figure 19 provide a summary of the results obtained on each of the islands for this experiment.

Table 6. Results for coarse and fine islands for pop124_10f_10c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	9	-189	34	139	180.26
Coarse 1	9	-162	22	-208	183.35
Coarse 2	9	-162	22	-239	183.38

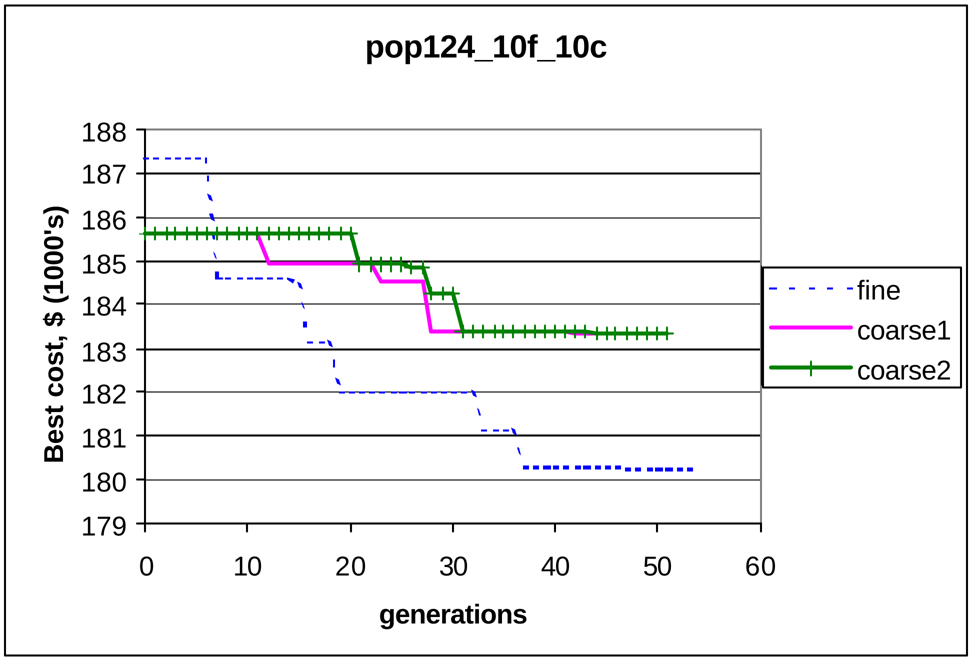


Figure 19. Best cost trend for all the islands for pop124_10f_10c

b) ***pop124_5f_10c***: This experiment also used a population size of 124 for each of the coarse islands and the fine island. Migration rate was 5% from coarse islands to the fine island, and 10% between the coarse islands. The job was submitted on NCSA's Origin 2000 dedicated queue, and 41 processors per island were used. The job ran on the Balder machine, and took approximately 19.8 hours of run time and 1686 hours of CPU time. Table 7 and Figure 20 provide a summary of the results obtained on each of the islands for this experiment.

Table 7. Results for coarse and fine islands for pop124_5f_10c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	8	-209	36	132	181.18
Coarse 1	9	-162	22	-208	183.35
Coarse 2	9	-162	22	-208	183.35

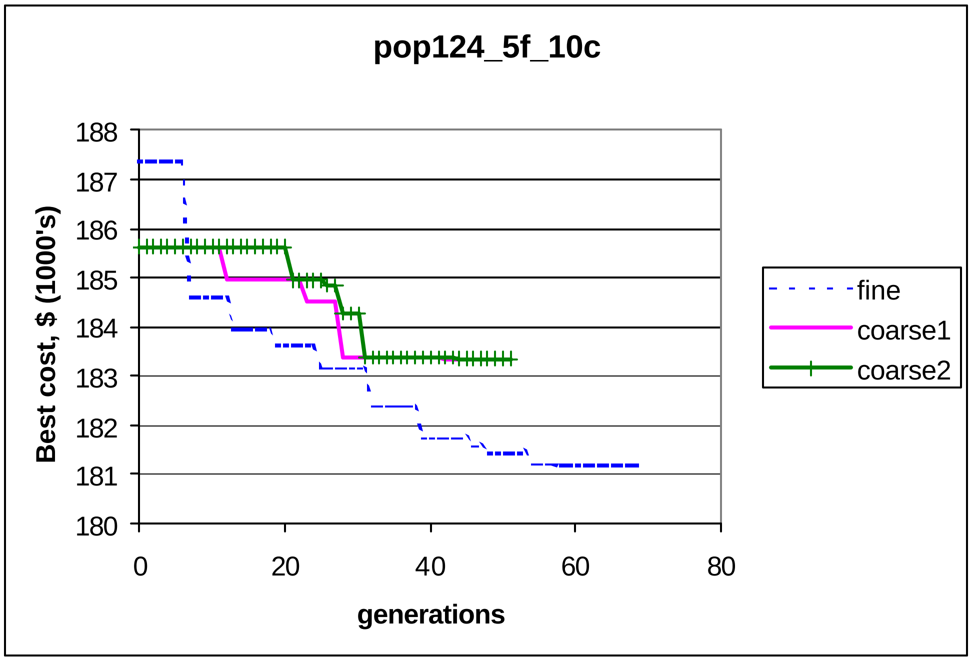


Figure 20. Best cost trend for all the islands for pop124_5f_10c

c) ***pop124_10f_5c***: This experiment used a population size of 124 for each of the coarse islands and the fine island. Migration rate was 10% from coarse islands to the fine island, and 5% between the coarse islands. The job was submitted on NCSA's Origin 2000 dedicated queue, and 41 processors per island were used. The job ran on the Balder machine, and took approximately 13.2 hours of run time and 382.3 hours of CPU time.

Table 8 and Figure 21 provide a summary of the results obtained on each of the islands for this experiment.

Table 8. Results for coarse and fine islands for pop124_10f_5c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	9	-189	34	139	180.26
Coarse 1	22	-212	9	-169	183.31
Coarse 2	22	-206	9	-164	183.31

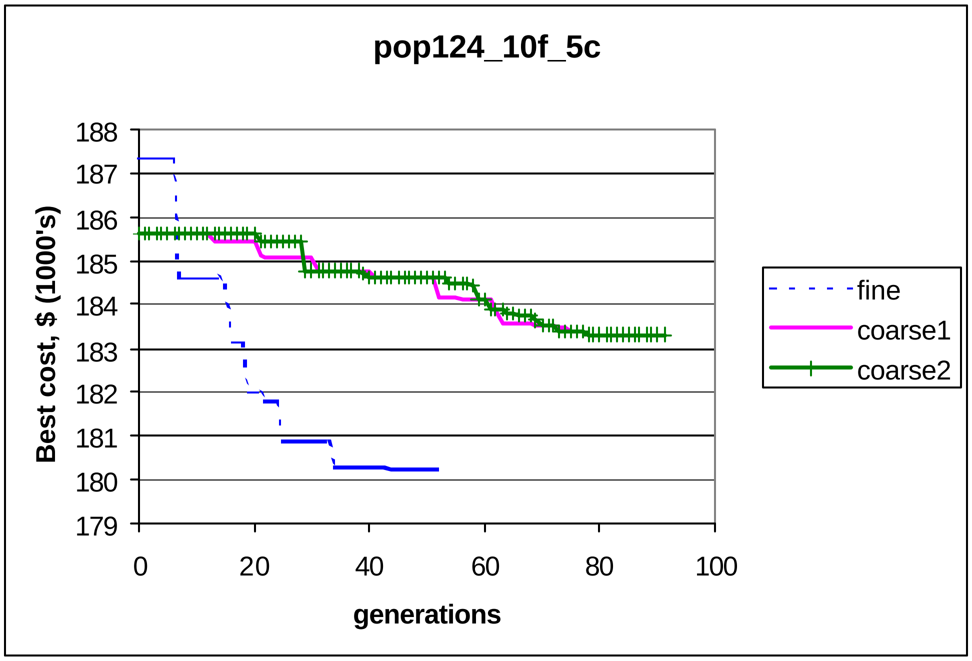


Figure 21. Best cost trend for all the islands for pop124_10f_5c

d) ***pop84_10f_10c***: This experiment used a population size of 84 for each of the coarse islands and the fine island. Migration rate was 10% from coarse islands to the fine island, and 10% between the coarse islands. The job was submitted on NCSA's Origin 2000 dedicated queue, and 41 processors per island were used. The job ran on the Balder machine, and took approximately 10 hours of run time and 571.8 hours of CPU time. Table 9 and Figure 22 provide a summary of the results obtained on each of the islands for this experiment.

Table 9. Results for coarse and fine islands for pop84_10f_10c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	15	-173	22	-132	182.51
Coarse 1	9	-182	22	-192	184.87
Coarse 2	9	-183	22	-192	184.79

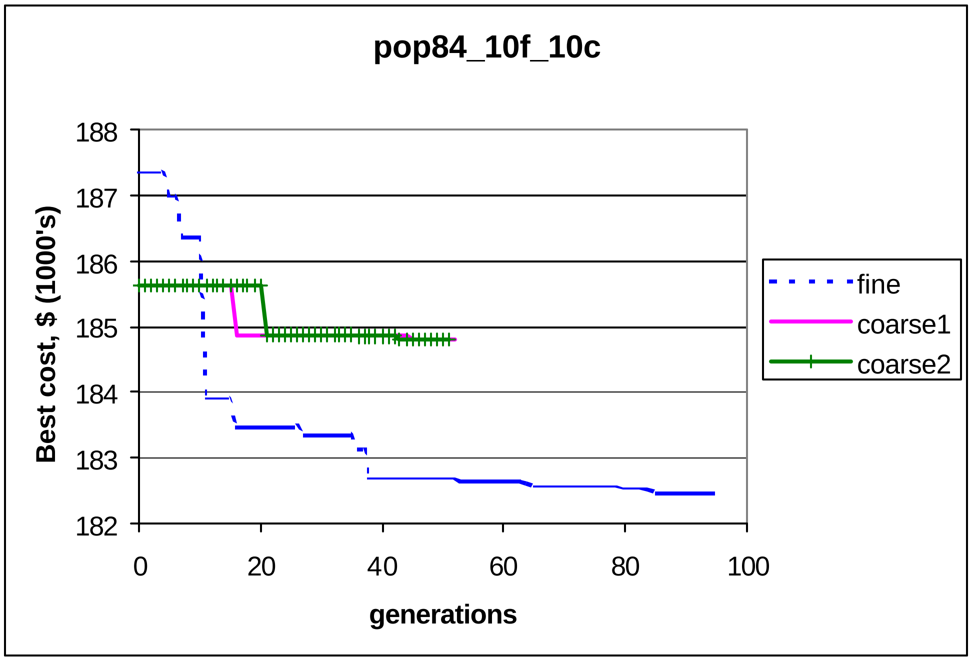


Figure 22. Best cost trend for all the islands for pop84_10f_10c

e) ***pop84_5f_10c***: This experiment used a population size of 84 for each of the coarse islands and the fine island. Migration rate was 5% from coarse islands to the fine island, and 10% between the coarse islands. The job was submitted on NCSA's Origin 2000 dedicated queue, and 41 processors per island were used. The job ran on the Forseti1 machine, and took approximately 26 hours of run time and 1338.6 hours of CPU time. Table 10 and Figure 23 provide a summary of the results obtained on each of the islands for this experiment.

Table 10. Results for coarse and fine islands for pop84_5f_10c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	15	-136	22	-171	182.66
Coarse 1	9	-180	22	-192	184.79
Coarse 2	9	-183	22	-192	184.79

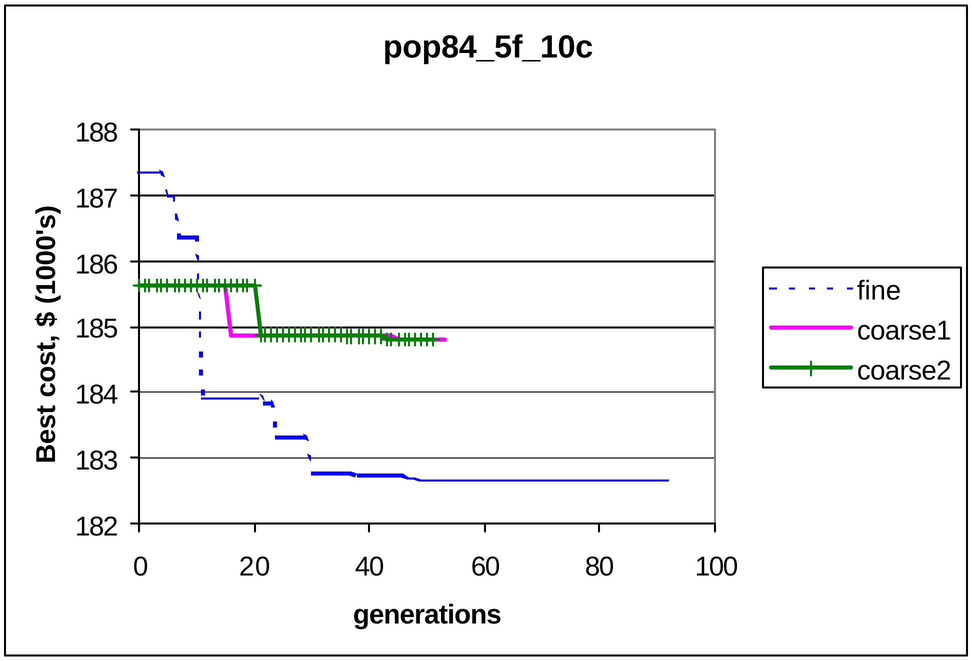


Figure 23. Best cost trend for all the islands for pop84_5f_10c

f) ***pop84_10f_5c***: This experiment used a population size of 84 for each of the coarse islands and the fine island. Migration rate was 10% from coarse islands to the fine island, and 5% between the coarse islands. The job was submitted on NCSA's Origin 2000 dedicated queue, and 41 processors per island were used. The job ran on the Balder machine, and took approximately 5.5 hours of run time and 335 hours of CPU time. Table 11 and Figure 24 provide a summary of the results obtained on each of the islands for this experiment.

Table 11. Results for coarse and fine islands for pop84_10f_5c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	15	-147	22	-158	182.53
Coarse 1	15	-190	22	-192	185.45
Coarse 2	15	-190	22	-189	185.45

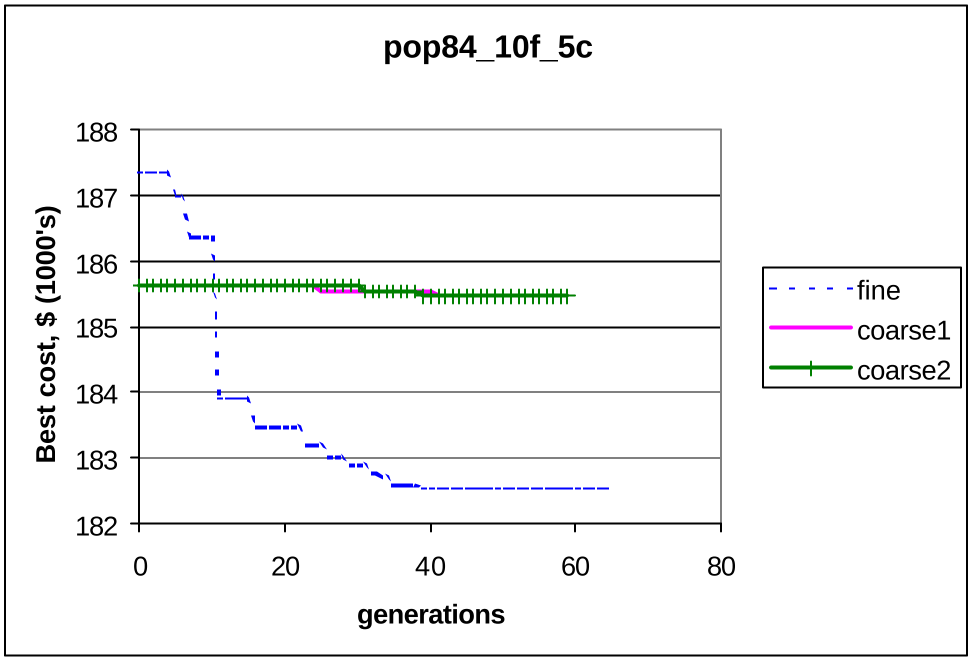


Figure 24. Best cost trend for all the islands for pop84_10f_5c

g) ***pop40_10f_10c***: This experiment used a population size of 40 for each of the coarse islands and the fine island. Migration rate was 10% from coarse islands to the fine island, and 10% between the coarse islands. The job was submitted on NCSA's Origin 2000 st_lj queue, and 10 processors per island were used. The job ran on the jord1 machine, and took approximately 10.9 hours of run time and 67.4 hours of CPU time. Table 12 and Figure 25 provide a summary of the results obtained on each of the islands for this experiment.

Table 12. Results for coarse and fine islands for pop40_10f_10c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	23	-137	20	-178	182.31
Coarse 1	4	-188	22	-209	183.84
Coarse 2	4	-188	22	-209	183.84

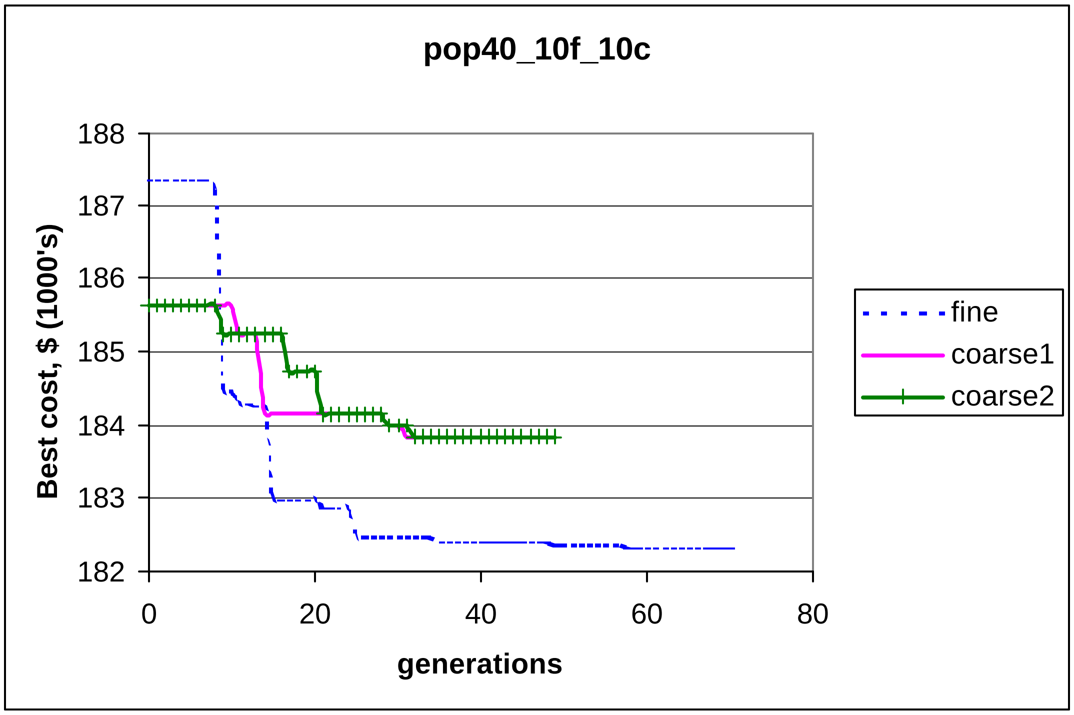


Figure 25. Best cost trend for all the islands for pop40_10f_10c

h) ***pop40f-84c_5f_5c***: This experiment used a population size of 40 for the fine island and 84 for the coarse islands. Migration rate was 5% from coarse islands to the fine island, and 5% between the coarse islands. The job was submitted on NCSA's Origin 2000 st_lj queue, and 10 processors per island were used. The job ran on the huldra machine, and took approximately 10 hours of run time and 66 hours of CPU time. Table 13 and Figure 26 provide a summary of the results obtained on each of the islands for this experiment.

Table 13. Results for coarse and fine islands for pop40f-84c_5f_5c

Island	Well 1 location	Well 1 pumping rate (m^3/day)	Well 2 location	Well 2 pumping rate (m^3/day)	Best Cost, $(1000's)
Fine	23	-151	20	-161	182.31
Coarse 1	15	-200	17	-175	183.84
Coarse 2	15	-231	21	-237	183.84

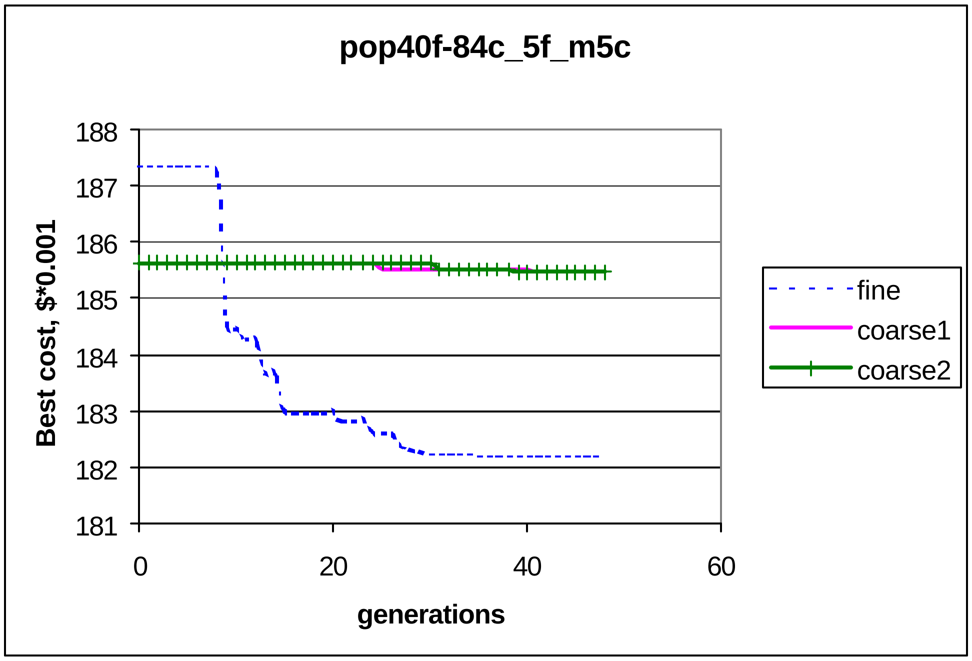


Figure 26. Best cost trend for all the islands for pop40f-84c_5f_5c