MULTISCALE ISLAND INJECTION GENETIC ALGORITHMS FOR GROUNDWATER REMEDIATION

BY

EVA SINHA

B-Tech, Indian Institute of Technology, Kanpur, 2002

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Civil and Environmental Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

ABSTRACT

Genetic algorithms have been shown to be powerful tools for solving a wide variety of water resources optimization problems. Applying these approaches to complex, large-scale water resources applications can be difficult due to computational limitations, especially when a numerical model is needed to evaluate different solutions. This problem is particularly acute for solving field-scale groundwater remediation design problems, where large grids are often needed for accuracy. Finer grids usually improve the accuracy of the solutions, but they are also computationally expensive. Multiscale parallel genetic algorithms have been shown to improve the performance of engineering design problems that use spatial grids. In this thesis we present multiscale island injection genetic algorithms (IIGAs), in which the optimization algorithms have different multiscale populations working on different islands (groups of processors) and periodically exchanging information. The approach is tested using a field-scale pump-and-treat design problem at the Umatilla Army Depot. The performance of this approach is compared with the results of a simple genetic algorithm implemented on a single population with both fine and coarse grids, a single population uniform multiscale approach, and another island injection approach developed previously. The results show that the new approach is a substantial improvement over existing approaches for this application. The approach found the same solution as much as 83% faster than the simple genetic algorithm and 15-52% faster than other previously formulated multiscale strategies. These findings indicate substantial promise for multiscale parallel approaches to improve solution of complex water resources applications at the field scale.

To my Family

ACKNOWLEDGMENTS

First of all I would like to thank my Advisor Dr. Barbara Minsker for providing me with her guidance and support during my Master's work. She was always there to encourage and motivate me during tough times and to provide me with the right insight at times of dilemma. Overall she has been a great mentor and guide and it has been an honor to work with her for the past two years.

I would also like to thank the various faculty members in the University of Illinois at Urbana-Champaign and at the Indian Institute of Technology, Kanpur under whom I took several courses, and learned quite a bit about Civil and Environmental Engineering. They contributed a lot in the making of the person that I am today.

I would also like to acknowledge the U. S. Army Research Office under grant number DAAD19-001-1-0025 for funding this research.

I would also like to thank my group mate Shengquan Yan for helping me in getting my runs complete. Without his help the thesis completion would have been extremely difficult. I would also like to thank Meghna Babbar for helping me with my thesis as well as for being a great friend at all times. At this point I would also like to thank all the rest of my friends (Gautam, Mukesh, Chhavi, Manuj, Amit, Shivi and the list goes on…) who were always there for me and have been very supportive throughout.

Last but not the least I would like to thank my family for having their trust in me and for always giving me the right to do and achieve what I aimed for in my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Groundwater is one of the major water resources in the United States and accounts for 50% of all drinking water supplies in the country. It is also the major source for providing water for irrigation. However, various pollutants can contaminate groundwater such as pesticides and fertilizers and toxic substances from waste sites, mining sites, and road salts, which can easily seep through the soil layer and reach the groundwater. Leaks from storage tanks containing gasoline, oil or other chemical, septic tanks, landfills or hazardous waste sites are also common groundwater pollutants. The National Research Council (1994) estimated that in the United States there are approximately 400,000 contaminated groundwater sites and the cost of cleanup of these sites is estimated to be around $1 trillion.

The most frequently used approach for groundwater remediation is pump-and-treat technology, which involves extracting the contaminated water from the aquifer, treating it to remove the contaminants, and injecting the water back into the aquifer. However, the installation, operation, and maintenance cost of these pump-and-treat systems is very expensive and hence it is imperative that the right design alternative be chosen for cleanup. To assist in the design process, numerical models are often developed to predict the fate and transport of contaminants using different design alternatives. By using optimization models in combination with these numerical models, one can determine the best combination of well locations and pumping rate and thus identify more cost efficient groundwater remediation designs. Many studies, including *Ritzel et al* (1994), *Karatzas*

and Pinder (1996), *McKinney and Lin* (1996), *Rizzo and Dougherty* (1996), *Culver and Shoemaker* (1997), *Minsker and Shoemaker* (1998), *Aly and Peralta* (1999), *Chan Hilton and Culver* (2000), *Smalley et al* (2000), *Erikson, et al* (2002), *Maskey et al.,* (2002) and *Liu and Minsker* (2002) have applied this type of approach for groundwater remediation management.

The degree of accuracy of the numerical models used for optimal design is a function of several factors, including the size of numerical grid used in the simulation model. Fine grids usually improve accuracy but can increase computational effort substantially. Coarse grids, though computationally less intensive, tend to produce less reliable solutions because of the numerical inaccuracies in evaluating potential designs (*Babbar et, al* 2003). Recently, some water resources optimization work has shown improved performance by combining coarse and fine grids in a multiscale approach (*Babbar.,* 2002; *Liu and Minsker,* 2001, 2002 and 2004). *Liu and Minsker* (2001, 2002, and 2004) used multiscale approaches to obtain optimal in situ bioremediation designs for a hypothetical test case using a derivative based optimization method called SALQR. *Babbar* (2003) tested two different multiscale strategies on a hypothetical test case.

In this thesis, the work of *Babbar* (2003) is extended to more thoroughly consider a multiscale approach called Island Injection Genetic Algorithm (IIGA). The IIGA (Island Injection Genetic Algorithm) approach has been shown to out perform the simple genetic algorithm method in several studies in other fields (*Tanese et al., 1987; Eby et al., 1997 and Lin et al., 1997*). *Babbar* (2003) tested one type of IIGA on a hypothetical

groundwater remediation design test case and obtained poor performance on a hypothetical test case. In this thesis we develop a new IIGA approach and rigorously test both the new approach and several previous multiscale approaches on a field-scale groundwater remediation optimization application at Umatilla Army Depot.

The thesis is organized as follows. First the Umatilla Army Depot site and groundwater remediation optimization formulation are described. Next, the multiscale genetic algorithms tested in this work are discussed in detail. A brief overview of the distributed parallel computation approach used for solving the genetic algorithms is also provided. The next chapter shows the results obtained with the different approaches on the Umatilla case. The final chapter provides a brief conclusion of the entire work.

2. UMATILLA CASE STUDY

The multiscale strategies, described in detail in the next section, were tested on a field-scale groundwater remediation design case study at Umatilla Army Depot. This chapter describes the Umatilla site and gives details on the mathematical formulation used for optimizing pump-and-treat design at the site.

2.1 Description of the Umatilla Chemical Depot Remediation Site

Umatilla Chemical Depot is a 19,728-acre military reservation located in northeastern Oregon. It was established in 1941 as an ordnance depot for the storage and handling of munitions. From the 1950s until 1965, the depot operated an onsite explosives washout plant. During the 15 years of operation as a washout plant, an estimated 85 million gallons of wash water was discharged to two unlined lagoons. This wash water infiltrated into the soil and carried the explosives to a shallow, unconfined aquifer located approximately 47 feet under the depot surface. Because of soil and groundwater contamination, the site was placed on EPA's National Priorities List (NPL) in 1984.

Two major contaminants present in the aquifer are *2,4,6 Trinitrotoluene* (TNT) *and Hexahydro-1,3,5-trinitro-1,3,5-triazine*(RDX), which are the primary cleanup targets. A pump-and-treat system was designed by the U.S. Army Corps of Engineers (USACE, 1996 and 2000) to contain and remove the RDX and TNT plumes. Figure 2.1 shows the existing pump-and-treat system, which consists of four extraction wells (three active) and three injection basins to return the extracted water to the aquifer after treatment. The

contaminated water is treated using granular activated carbon columns. This design was created using numerical models to predict the groundwater flow and contaminant transport. The numerical models were made by USACE using MODFLOW 2000 (*Harbaugh et al.*, 2000) and MT3DMS (*Zheng and Wang*, 1999). Figure 2.1 shows the modeled RDX and TNT plumes as of October 2000. As shown in the figure the RDX plume is much larger than the TNT plume because TNT is more strongly absorbed to the aquifer materials.

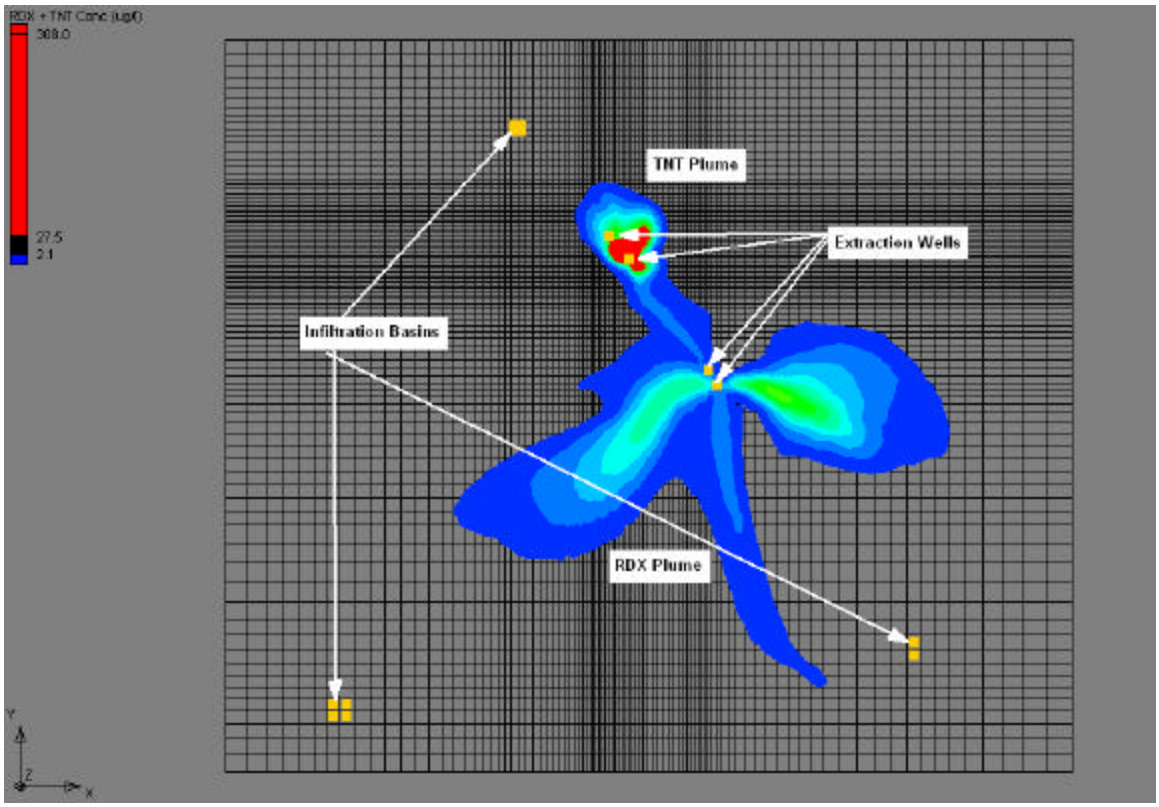

FIGURE 2.1 Existing RDX and TNT plumes at the site as of October 2000, along with the pumping well and infiltration basins. Source ESTCP (2003)

The numerical grid used in the USACE modeling effort is shown in Figure 2.1. The entire site was divided into 5 model layers with 132 rows and 125 columns (this grid size

is later on referred to as the fine grid). The first layer represents the shallow aquifer of concern, which is an alluvial aquifer, and the last four layers are in a silt and weathered basalt unit, and are included only to improve predictions in the shallow aquifer. The alluvial aquifer was divided into approximately 12 homogenous hydraulic conductivity zones, with hydraulic conductivities ranging from 10^{-6} to 10^{-3} m/s.

The coarse grid was derived from the fine grid by merging consecutive rows and columns using GMS (Groundwater Modeling System) version 3.1. The coarse grid has 66 rows, 64 columns and 5 layers. Figure 2.2 depicts the plume on the coarse grid.

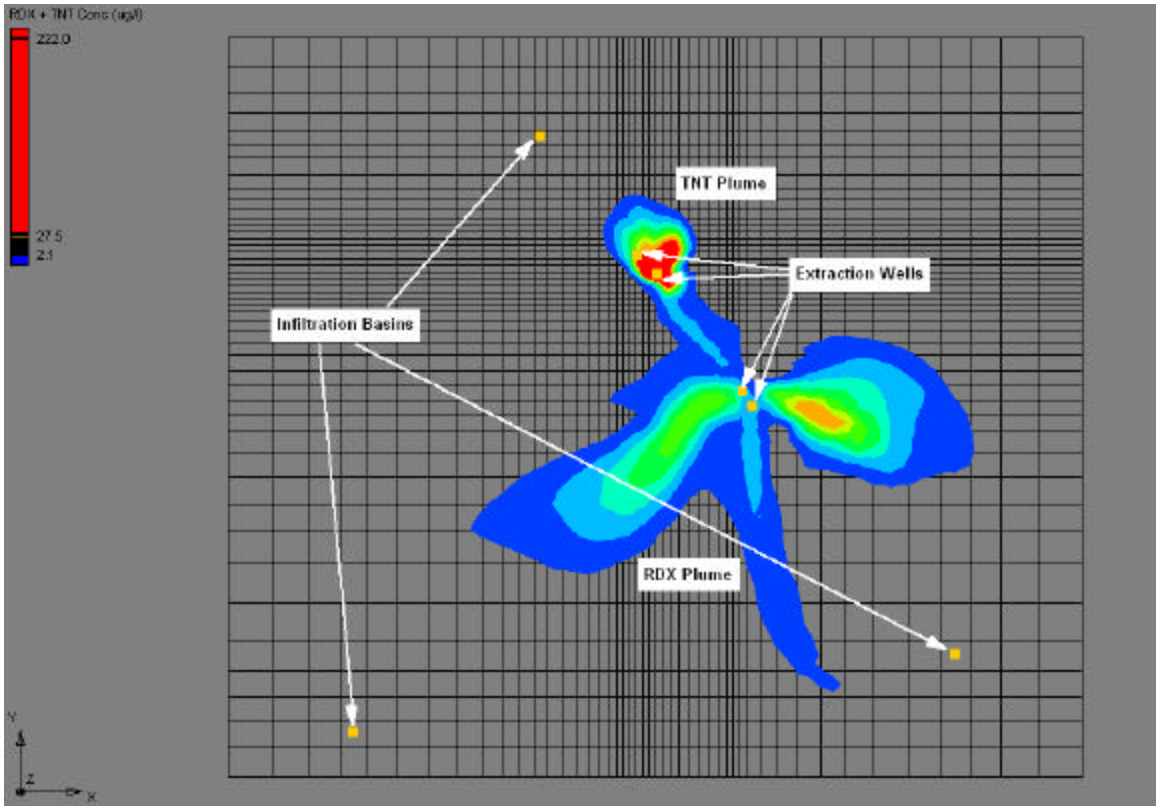

Figure 2.2 Representation of the plume on the coarse grid

2.2 Optimization Formulation

The Umatilla site was chosen by an Environmental Security Technology Certification Program (ESTCP) project as one of three demonstration sites for transport optimization of pump-and-treat systems (*Minsker et al.,* 2003). Three optimization formulations were developed for each site in that project. Our research is focused on the first formulation for the Umatilla site, which identifies locations and pumping rates of extraction wells and injection basins to minimize the total remediation cost. The detailed formulation equations are quite extensive and only a summary is given here; see Minsker et al. 2003 for details.

The objective function of Formulation 1 can be summarized as follows:

Minimize: CCW + CCB + FCL + FCE + VCE + VCG + VCS (2.1)

where:

CCW: Capital costs of new wells

CCB: Capital costs of new recharge basins

FCL: Fixed cost of labor

FCE: Fixed costs of electricity

VCE: Variable electrical costs of operating wells

VCG: Variable costs of changing GAC units

VCS: Variable cost of sampling

The various constraints are:

$Q_1 + Q_2 = 1170$ gpm (2.2)

$Q_1 = 360$ gpm (2.3)

$Q_2 = 900$ gpm (2.4)

$C_{RDX} = 2.1$ µg/l (2.5)

$C_{TNT} = 2.8$ µg/l (2.6)

Total injection = total extraction (2.7)

where: Q_1 and Q_2 are the total pumping rates from two different hydrogeological zones identified in the area of study and C_{RDX} and C_{TNT} are the RDX and TNT concentrations.

The specific conditions used for the design are a maximum of four new extraction wells, three new injection basins and a maximum cleanup time of five years. The five-year limit was imposed to reduce computational effort associated with testing the multiscale approaches, since the optimal solutions found in the ESTCP project were all less than five years (*Minsker et al.*, 2003).

To solve this formulation using a genetic algorithm, each remediation design was represented by a string of binary numbers. These binary numbers represent different elements of the design: pumping well and infiltration basin location (for new wells and basins only), pumping well rates (for both new and old wells), and installation flags for wells and infiltration basins. To reduce computational complexity, the infiltration rates are selected by imposing mass balance between extraction and infiltration, assuming that all of the water extracted is divided evenly among all infiltration basins in use. Thus the total number of decision variables is 29 and the string representing each design has 131 bits, as follows:

- Locations of 4 new wells and 3 basins, where each well or basin is allowed to be sited in one of 32 (represented by 5 bits) or 53 (represented by 6 bits) locations on the grid;
- 7 installation flags for the 4 new wells and 3 basins, represented by one bit each;
- 7 flags for use of the 4 existing wells and 3 existing basins, represented by one bit each; and
- 8 pumping rates for the 4 new and 4 existing wells, each represented by 10 bits for each well.

3. METHODOLOGY

This chapter briefly describes the simple and multiscale genetic algorithms (GAs) used in this work. At the end of the chapter details are provided on the parallel computational environment used to solve all of these approaches for the Umatilla case.

3.1 Simple Genetic Algorithms

Simple genetic algorithms (SGAs) are search algorithms that evolve a population of solutions to the optimal solution(s) using principles of natural selection ('survival of the fittest') with three basic operators: reproduction, crossover and mutation. Genetic algorithms are theoretically and experimentally proven to provide robust search in complex spaces (*Holland, 1975*). Their main advantage over conventional gradient based methods lies in their ability to solve discrete, non-convex and discontinuous problems (*Goldberg, 1989*), which enables easy incorporation of complex existing simulation models. Simple genetic algorithms (SGAs) have also been shown to provide good solutions to various groundwater remediation design problems [*McKinney and Lin,* 1994; *Ritzel et al.,* 1994; *Wang and Zheng,* 1998; *Aly and Peralta,* 1999a, 1999b; *Smalley et al.,* 2000; *Chan Hilton and Culver,* 2000, 2001; *Maskey et al.,* 2002; *Erikson et al.,* 2002; and *Gopalakrishnan et al., 2003*].

The first step of the simple genetic algorithm is the creation of an initial random population of possible remediation design alternatives. In a binary SGA, used in this work, each design alternative is represented by a string of binary digits. Each member of

the population ("individual") is evaluated for "fitness" (performance on the objective function and constraints, which are incorporated using a penalty function). Then a new population ("generation") is created using three basic operators, which in this work are binary tournament selection, uniform crossover, and simple mutation (see Goldberg 1989 for details). In addition, $\mu+\lambda$ selection scheme is applied in this work, where μ represents the parent population and λ the child population. In this selection mechanism, the next generation is created from the μ best individuals in the total population of $\mu+\lambda$ individuals created after the other 3 operations. These operations are continued until the population converges to the optimal or near-optimal solution, in this case when 90% of the population is the same or the maximum number of generations is reached. The guidelines of *Reed et al.* (2000) were followed to set the population size to 160, the maximum number of generations to 262, the probability of crossover to 0.5, and the probability of mutation to $1/N$, where N is the population size.

3.2 Multiscale Strategies

Island injection genetic algorithms (IIGAs) are a multiscale version of "multiple-deme" parallel GAs. Multiple-deme GAs use multiple subpopulations ("islands"), each of which performs independent GA runs and exchanges its best individuals with the other populations after a few generations. The IIGA approach uses multiple subpopulations operating at different spatial scales and has been shown to outperform the simple genetic algorithm method in applications in other fields (*Tanese et al., 1987; Eby et al., 1997 and Lin et al., 1997*). The islands can be connected using different topologies: bidirectional rings, unidirectional rings, hypercubes, tree structure, etc. For example, *Punch* (1995) and

Goodman (1997) used a tree-like IIGA topology for the optimization of composite structures, where exchange of information was only from a low resolution node (having lesser design elements and thus a smaller search space) to a high resolution node (having large number of design elements and thus a larger search space). There can also be different migration rates (the percentage of individuals being migrated) and different migration intervals (number of generations after which migration takes place) between various islands.

Babbar (2002) used an IIGA that consisted of three islands, two of them working with the coarse grid and one on the fine grid (Figure 3.1). The islands working on the coarse grid periodically injected their best solutions into the fine grid population while also exchanging solutions among themselves after a specific number of generations. Two different migration rates (the best 5% and 10% of the population) were tested and it was observed that the higher migration rate (10%) performed better that the low migration rate (5%). However, this approach produced results for a hypothetical test case that were much more computationally intensive than the SGA using the fine grid alone. The hypothesis of this thesis is that the failure was either because the test case could be solved relatively quickly even on the fine grid or because of inefficiencies in load balancing among the parallel processors when 2 of the 3 populations are using the fast coarse grid and one population is using the slower fine grid. This IIGA approach will hereafter be referred to as the IIGA(NUM), due to the non uniformity of populations on different islands. In this thesis, this approach will be compared with a more uniform multiscale approach on the Umatilla case, which is much more computationally intensive than the

hypothetical test case, in order to identify what caused the failure of this approach previously.

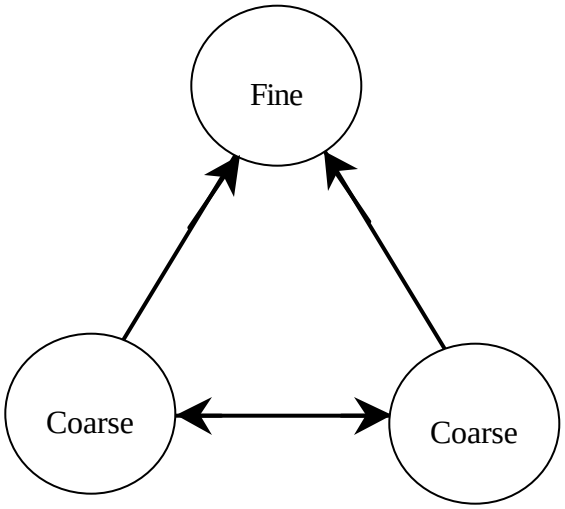

Figure 3.1 Island Injection Topology with Non Uniform Multiscale Approach [IIGA(NUM)] (Babbar 2002)

The new island injection strategy proposed here consists of three islands (depicted in Figure 3.2), each searching for the optimal solution using a 'uniform multiscale strategy', hereafter referred to as IIGA(UM). The uniform multiscale strategy was formulated for a single population by *Babbar et al* (2003). The approach is to first use the coarse grid numerical model for an initial evaluation of the entire population. Then, a specified percentage of the best individuals in the population are evaluated on the fine grid. For crossover, tournament selection uses a biased technique to select the individual for mating. This biased technique prefers individuals tested on the fine grid over individuals tested on the coarse grid, even if the apparent fitness of individual tested on the coarse grid is better than that tested on the fine grid. The proportion of individuals selected from the fine grid group in the population is slowly increased to allow the individuals tested on the fine grid to take over the population after sufficient exploration using the coarse grid

numerical model. This approach was shown to be significantly faster than the SGA on a hypothetical test case with a single population (*Babbar et al* 2003).

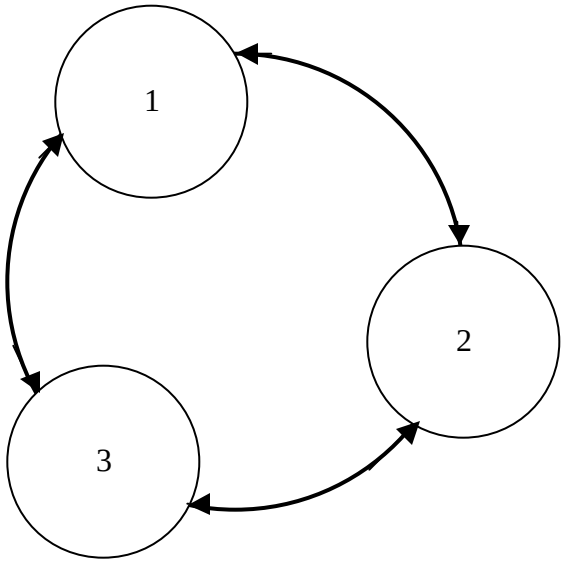

Figure 3.2 Island Injection Topology with Uniform Multiscale Approach [IIGA(UM)]

This single population uniform multiscale strategy is implemented on all three islands to create the IIGA(UM) strategy. All the islands have different starting populations and use the uniform multiscale strategy with the same percentage of individuals tested on the fine grid. The islands exchange their best individuals at the higher resolution (fine scale) after a fixed number of generations. Best individuals at fine scale migrate from one island to other and replace the worst individuals at the same scale there. Though transferring of individuals should result in exchange of good building blocks, excessive migration leads to premature convergence (Cantu'-Paz, 1999). Babbar (M.S thesis 2002) tested two migration rates (5% and 10% of the population being migrated) on a test case and observed that the high migration rate resulted in better solution quality. Thus in our results we fixed the migration rate at 10% and varied the frequency of migration. Two

different frequencies, exchange after every 5^{th} generation and exchange after every 10^{th} generation, are tested to observe their effect on performance.

3.3 Distributed Parallel Computation.

Parallel computation on a distributed standard office network cluster system was used for performing the runs to test the approaches described above. This was possible because the fitness function evaluation of each individual in a generation is independent and thus these evaluations can be distributed through the network and run simultaneously on multiple host computers.

The parallel system consists of a central client desktop computer running the GA portion of the code and multiple service computers running the numerical models. On each service computer, a process (i.e., fitness evaluation) is running in the background (it keeps running whether or not the computer is locked, logged in or logged off). When the code starts, the connection is initialized between the client and the service computers to exchange the necessary supporting files. At the beginning of each generation the client sends the members of the population to the service computers. The service computers simultaneously execute the numerical models after receiving the strings and then send the fitness values back. On the service computers these processes (fitness function evaluations) run in the background. The client continues the GA run after receiving all of the fitness function evaluations. Figure 3.3 shows the diagram for the distributed system structure.

Most of the service computers are interactive desktop computers on a standard office network. These computers can be shut down or rebooted by the user anytime. The distributed system is designed to tolerate such disturbances. It can sense the service computer's status and behave politely according to the following conditions:

- If a user is logging onto the computer, the service computer will suspend the numerical model or put it at a lower priority.
- If a user is logging off, locking the computer, or the screen saving program is running, the service computer will resume the numerical model with normal running priority.
- Each time when the service computer is rescheduling the numerical model, it sends feedback to the client. The client waits for a reasonable time. If the service computer is still busy, the client reroutes the fitness evaluation to another idle service computer.
- Whenever a service computer is shutting down, the client detects it and reroutes the fitness evaluation to another idle service computer.

Following these rules, the distributed network system effectively uses the idle computers on the network without disturbing their interactive users.

Both the service computer and client programs were developed in C++ language using TCP/IP protocol for network communication. The service computer program was designed to cross multiple platforms, currently allowing it to be used in Windows, Unix and Linux systems.

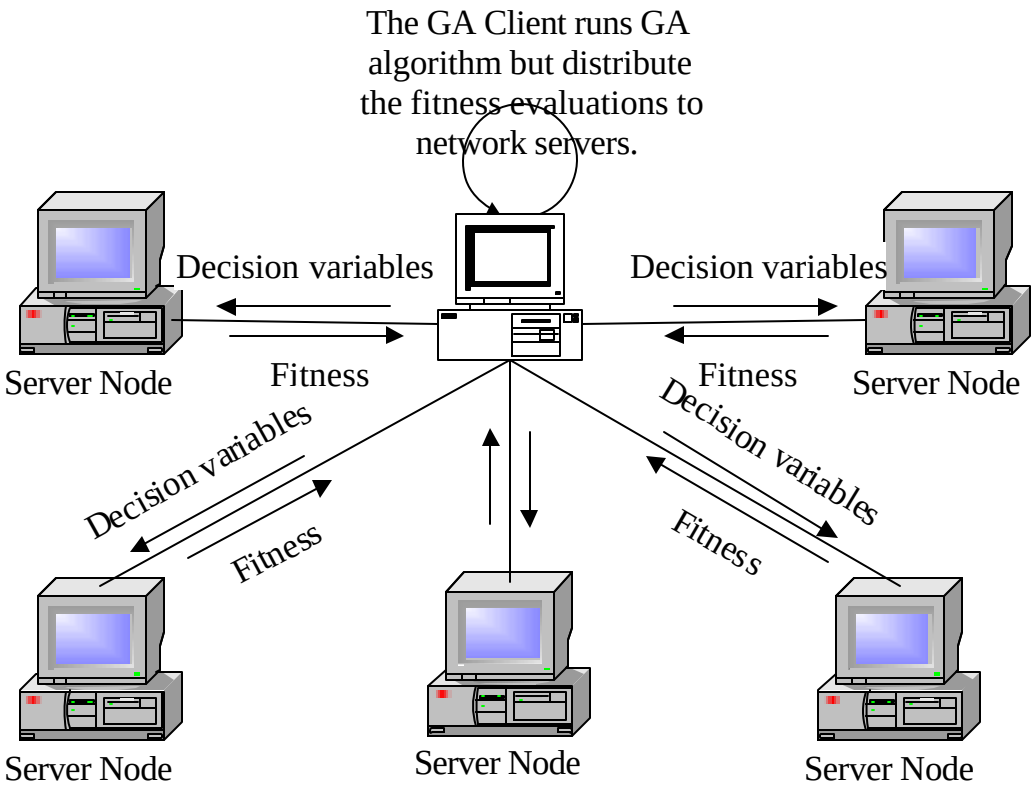

Figure 3.3: Distributed Cluster Structure

The IIGA (UM as well as NUM) scheme is implemented on this cluster by maintaining three different populations (three islands). The client computer runs one generation for the first island, then performs the run for the same generation on the second island, and finally on the third island. At the end of any generation, a check is made to see if individuals have to be migrated (i.e. if this is the next generation for migration to take place) and, if so, then the migration and population update takes place. Then the client starts the next generation and similarly moves from one island to another. As the client contains all of the data (i.e., three different populations and their fitness values), there is no computation time taken for migration of individuals. This scheme is also illustrated with the help of a flow chart in Figure 3.4.

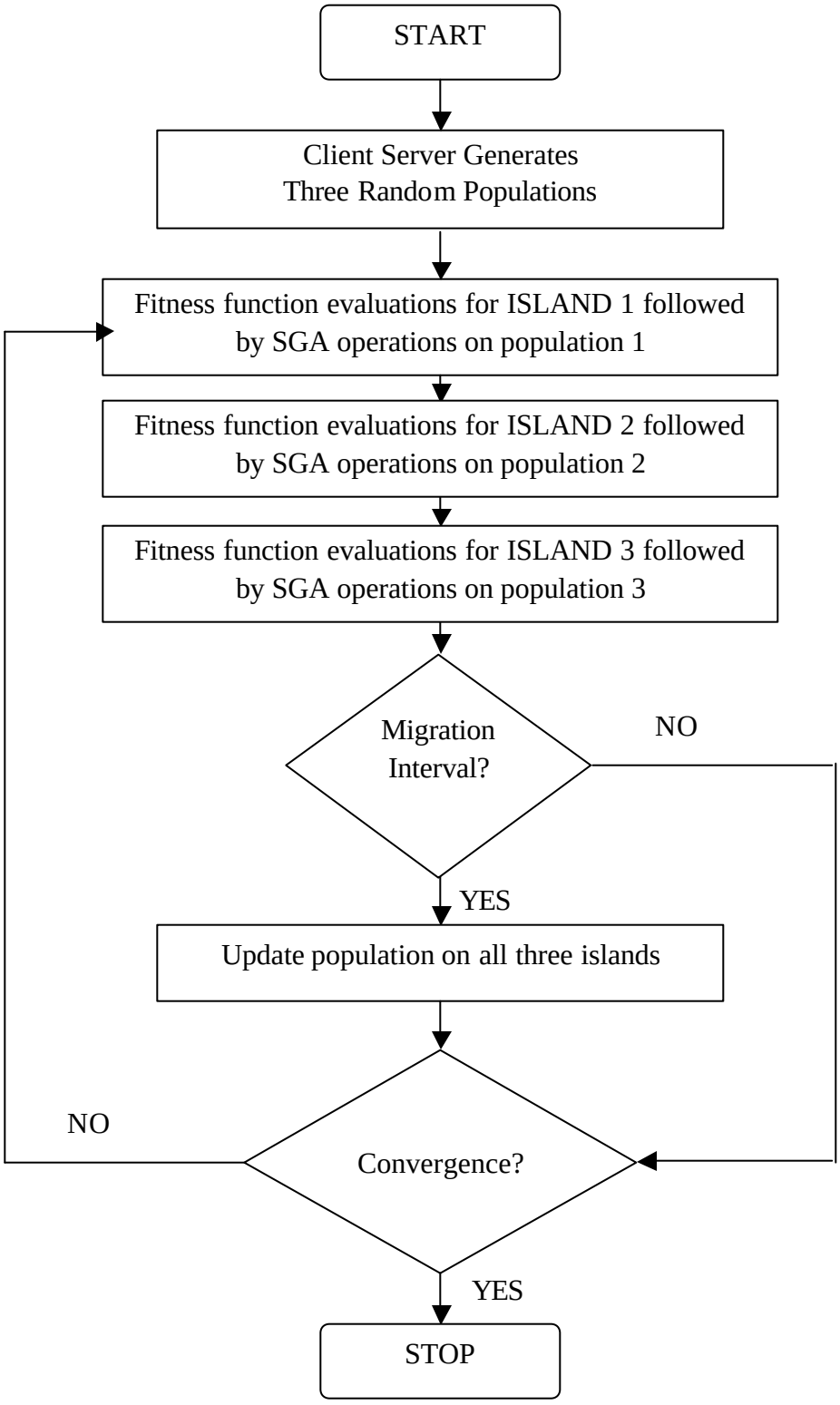

Figure 3.4 Flowchart depicting the implementation of IIGA(UM) and IIGA(NUM) methods on the parallel cluster.

4. RESULTS

In this section the results of applying the SGA and multiscale strategies to the Umatilla case are presented. All of the IIGA strategies had a fixed migration rate of 10%. The details of the various strategies tested along with the abbreviations used are shown in Table 4.1.

Table 4.1 Details of the various strategies tested on the Umatilla case.

Abbreviation	Population Distribution	Migration Interval	% Evaluated on the fine grid
SGA(F)	Single Population	NA	100%
SGA(C)	Single Population	NA	0% (Completely evaluated on the coarse grid)
MS_10%[1]	Single Population	NA	10%
IIGA(NUM)_5	Three subpopulations	5	One island evaluated completely on the fine grid and two on the coarse grid
IIGA(NUM)_10	Three subpopulations	10	One island evaluated completely on the fine grid and two on the coarse grid
IIGA(UM)_10%_5	Three subpopulations	5	10% (on all three islands)
IIGA(UM)_10%_10	Three subpopulations	10	10% (on all three islands)
IIGA(UM)_15%_5	Three subpopulations	5	15% (on all three islands)
IIGA(UM)_15%_10	Three subpopulations	10	15% (on all three islands)
IIGA(UM)_20%_5	Three subpopulations	5	20% (on all three islands)
IIGA(UM)_20%_10	Three subpopulations	10	20% (on all three islands)
IIGA(UM)_25%_5	Three subpopulations	5	25% (on all three islands)
IIGA(UM)_25%_10	Three subpopulations	10	25% (on all three islands)

Since the results may vary depending on the starting populations, three runs were completed for each strategy with three different starting populations.

[1] On separate runs performed on the Umatilla Case using the uniform multiscale strategy on a single island, it was observed that the 10% approach (where 10% of the population is evaluated on the fine grid) performed better than the 15% uniform multiscale approach and hence our results are compared only with the 10% approach.

Figure 4.1 shows the average and range in the number of fitness function evaluations on different grids across the 3 different starting populations. From the results we can see that the average number of fitness function evaluations on the fine grid decreased drastically for the uniform IIGA approach [IIGA(UM)] as compared to the SGA on the fine grid [SGA(F)]. The uniform IIGA approach [IIGA(UM)] took 79-94% fewer fitness function evaluations on the fine grid than the fine grid SGA [SGA(F)]. Performance of the cases with higher percentages of the population evaluated on the fine grid [e.g., IIGA(UM)_25%] were substantially more variable than those lower percentages [e.g., IIGA(UM)_10%], however. The number of fine grid evaluations was also substantially reduced for the uniform multiscale approach on a single population [MS-10%] (92% fewer than the SGA) and the non-uniform IIGA approach [IIGA(NUM)] (74-92% fewer than the SGA).

Given that each approach had different numbers of fitness function evaluations on each grid size, a fair comparison of the approaches requires that computational times of each approach be compared. However, the runs were completed using heterogeneous client and server computers with different processor speeds, and the clients can also be interrupted when interactive users log on. Hence, actual processing times for each run cannot be compared. In order to estimate the time taken to evaluate fitness for each grid size, a small test case with a population size of 20 was run for 100 generations on a desktop computer (Pentium III, 600MHz and 256MB RAM) using both the fine and the coarse grid. The average time taken for a single fine grid fitness function evaluation was

300 s and a single coarse grid fitness function evaluation took 50 s. These average times were then used to convert the number of fitness function evaluations shown in Figure 4.1 to a relative computational time.

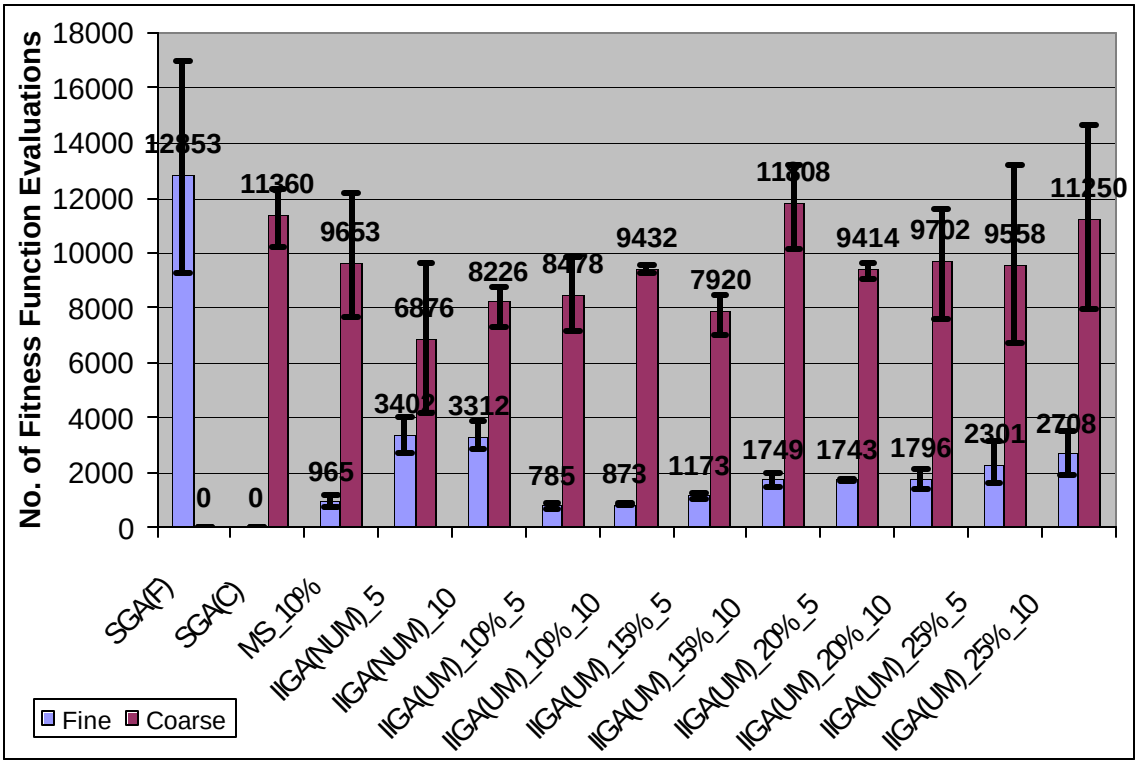

Figure 4.1: Average number of fitness function evaluations (along with maximum and minimum for different seeds) at each scale for the strategies listed in Table 4.1.

Figure 4.2 shows the average and range in relative computational time taken for the different approaches. From the figure we can observe that a migration interval of 5 generations performed better than that of 10 generations for both the uniform [IIGA(UM)] and non-uniform [IIGA(NUM)] IIGA strategies (3-33% better). From these results we can observe that a low migration interval (after 5 generations) improved performance by increasing the exchange of good "building blocks" (portions of good solutions that are combined in a genetic algorithm – see Goldberg 1989) relative to the high migration interval of 10 generations.

Of the different uniform IIGA [IIGA(UM)] strategies, the one with 10% of the individuals evaluated on the fine grid and a migration interval of 5 generations [IIGA(UM)_10%_5] performed the best. This strategy performed 12-44% faster than the other uniform IIGA strategies, 83% faster than the SGA on the fine grid [SGA(F)], 15% faster than the best uniform multiscale strategy with a single population [MS_10%], and 52% better than the best non-uniform IIGA [IIGA(NUM)] strategy. The uniform multiscale approach with 25% of the individuals evaluated on the fine grid [IIGA(UM)_25%_5 and IIGA(UM)_25%_10] performed the worst of the uniform IIGA approaches, apparently because the more accurate information gained by evaluating 25% of the population on the fine grid was not worth the added computational effort.

Here an important point to note is that in the parallel implementation used in this work there is no significant load balancing issue, where processors sit idle while waiting for others to finish their jobs, because jobs are simply redistributed among the many available processors when needed. Thus the non-uniform IIGA [IIGA(NUM)] required longer computational times than the uniform multiscale approaches [IIGA(UM) and MS_10%] only because the non-uniform approach evaluates an entire island on the fine grid as compared to only a small fraction of individuals being evaluated on the fine grid for the uniform approaches. This confirms that a uniform multiscale approach performs much better than the non-uniform IIGA approach. However when the uniform multiscale approach is combined with the IIGA approach (the various IIGA(UM) strategies), maximum computational time gain is achieved. This confirms a long-held principle in

evolutionary computation: favorable traits spread faster when small islands are used rather than a single large population (Cantu-Paz 2000).

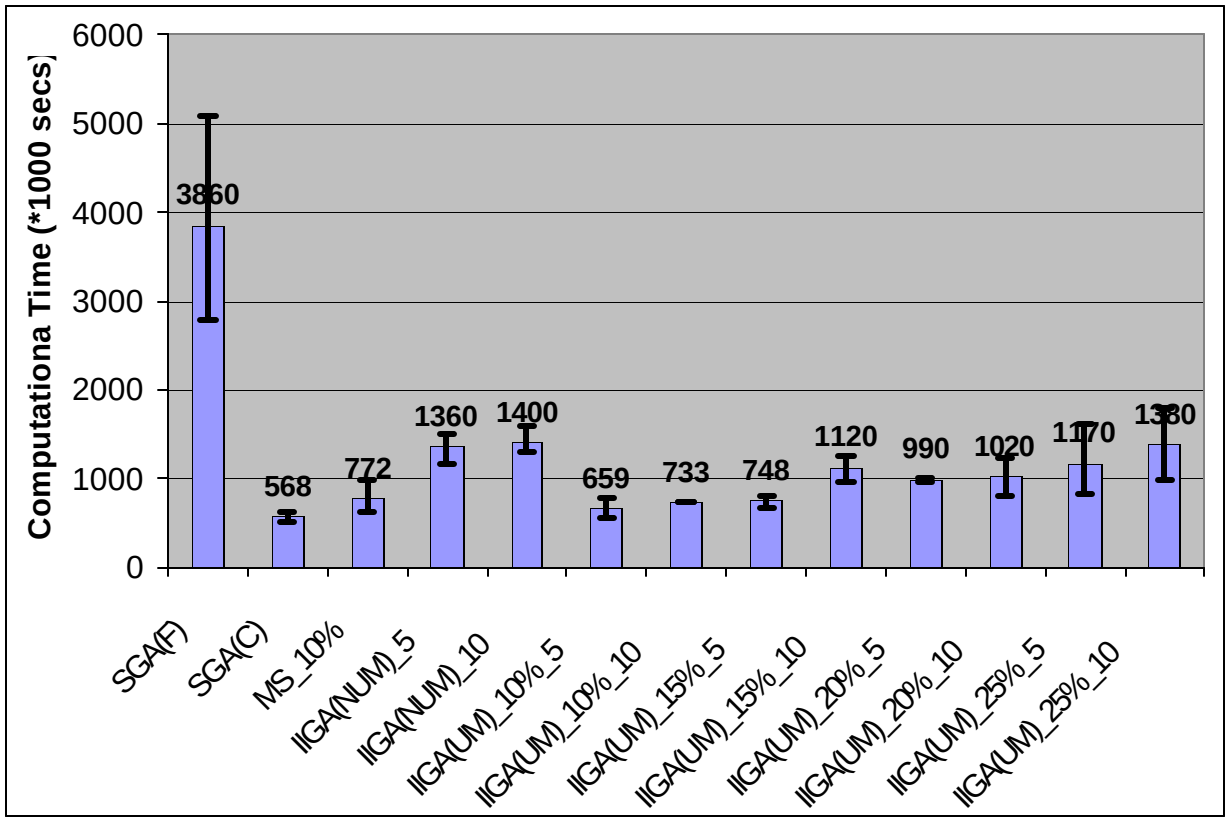

Figure 4.2: Relative computational time (along with maximum and minimum for different seeds) for each approach given in Table 4.1.

Figure 4.3 shows the average and range of optimal fitness values found by each approach in thousands of dollars. The optimal solution was $1.66 million dollars. This final fitness value was achieved by most of the strategies. The average SGA result with the coarse grid [SGA(C)] was 55% more costly than the optimal solution, which is a substantial difference particularly given the expense of the remediation. A few of the IIGA runs achieved somewhat inferior solutions, ranging from 1-9%, but approaches that performed the best for computational speed produced consistently reliable results.

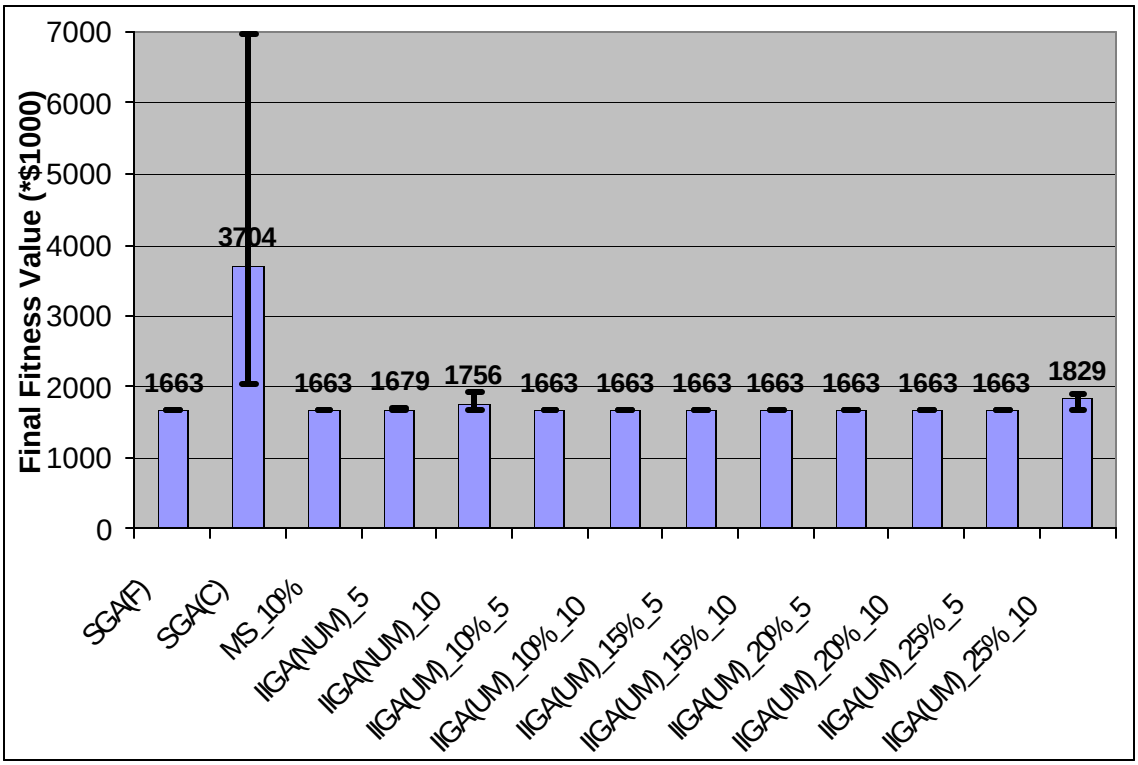

Figure 4.3: Average final fitness value (*1000$) (along with maximum and minimum for different seeds) for each approach given in Table 4.1.

5. CONCLUSIONS

In this work a new multiscale approach for water resources optimization was proposed and compared with the simple genetic algorithm and several previously proposed multiscale genetic algorithm approaches on a field scale application at Umatilla Army Depot. Overall, the new uniform multiscale approach with island injection GAs [IIGA(UM)] performed substantially better than the other multiscale approaches and the simple genetic algorithm. The best uniform multiscale IIGA strategy found the optimal solution 83% faster than the simple genetic algorithm on a single population using the fine grid only, 15% faster than the uniform multiscale strategy implemented on a single population only, and 52% faster than the non-uniform multiscale IIGA strategy. As expected, the simple genetic algorithm using the coarse grid only took the least amount of time compared with all other strategies, but it failed to find the correct solution and its final solution was 55% costlier than the optimal solution.

The new multiscale approach developed in this thesis offers hope for solving large-scale water resources applications that involve computationally intensive numerical models. Numerical models often require several hours for each run, which can create a significant challenge to optimization. For example, the third site included in the ESTCP transport optimization project (Minsker et al. 2003) required 2 hours per model run, even after maximizing the simulation model performance. Further testing is needed with additional starting populations (ongoing) and other applications to verify the performance of these approaches.

In this thesis, as in the ESTCP demonstration project from which the Umatilla case was obtained, uncertainty was not considered in order to reduce the computational effort of testing the different approaches. However, the uniform multiscale approaches tested in this work should provide even more benefits when uncertainty exists in the data, due to the substantial increase in the number of fitness function evaluations on the fine grid typically required for most uncertainty approaches to optimization with genetic algorithms (e.g., Smalley et al. 2000, Gopalakrishnan et al. 2003). Future work should verify this hypothesis, as well as test the performance of the uniform multiscale strategy formulated in this thesis on other field applications.

REFERENCES:

1) Aly, A. H., and Peralta, R. C. (1999a). "Comparison of a genetic algorithm and mathematical programming to the design of groundwater cleanup systems." *Water Resources Research,* 35(8), 2415-2425.

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

3) Babbar M. (2002) "Multiscale parallel genetic algorithms for optimal groundwater remediation design", M.S. Thesis, University of Illinois.

4) Babbar M., Minsker B. (2002) "A multiscale island injection genetic algorithm for optimal groundwater remediation" American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) 2002 Water Resources Planning & Management Conference.

5) Babbar, M., Minsker B., (2002) "A multiscale master-slave parallel genetic algorithm with application to groundwater remediation design", Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'2002.

6) Babbar, M., and Minsker B., (2003) "Multiscale strategies for solving water resources management problems with genetic algorithms" Environmental & Water

Resources Institute (EWRI) World Water & Environmental Resources Congress 2003 & Related Symposia.

7) Cantu'-Paz, E. (1998) "A survey of parallel genetic algorithms", *Calculateurs Paralleles, Reseaux et Systems Repartis,* Vol. 10, No. 2, pp. 141-171, Paris: Hermes.

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

9) Cantu'-Paz, E. (2000) *Efficient and accurate parallel genetic algorithms*, PhD thesis. Kluwer, Massachusetts, MA

10) Chan Hilton, A. B., and Culver, T. B. (2000). "Constraint handling for genetic algorithms in optimal remediation design." *Journal of Water Resources Planning and Management,* ASCE, 126(3), 128-137.

11) Chan Hilton, A. B., and Culver, T. B. (2001). "Sensitivity of optimal groundwater remediation designs to residual water quality violations." *Journal of Water Resources Planning and Management,* 126(3), 128-137.

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

13) Eby D., Averill R. C., Gelfand B., Punch W. F., Mathews O., Goodman E. D., (1997) "An injection island GA for flywheel design optimization" Invited Paper, Proc. EUFIT '97, - 5th European Congress on Intelligent Techniques and Soft Computing.

14) Erikson, M., Mayer, A., and Horn, J. (2002). "Multi-objective optimal design of groundwater remediation design systems: application of the niched Pareto genetic algorithm (NPGA)", *Advanced in Water Resources*, 25(1), 51-65.

15) Espinoza, F., B. S. Minsker, and D. E. Goldberg, " Performance evaluation and population reduction for a self-adaptive hybrid genetic algorithm (SAHGA)" Proceedings of the Genetic and Evolutionary Computation Conference: GECCO 2003, Part I, Springer, NY, ISBN 3-540-40402-6, 922-933.

16) Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, NY.

17) Goodman E. D., Averill R. C., Punch W. F., Eby D. J. (1997) "Parallel genetic algorithms in the optimization of composite structures", WSC2 Proceedings, World Wide Web Conference, www.egr.msu.edu/~ebydavid/ver5.html

18) Gopalakrishnan G., Minsker B., and Goldberg D.E., (2003) “Optimal sampling in a noisy genetic algorithm for risk-based remediation design”, *Journal of Hydroinformatics*, 5(1), 11-25.

19) Holland, J. H., (1975) *“Adaptation in natural and artificial systems”*. Ann Arbor: The University of Michigan Press.

20) Karatzas, G.P., and G.F. Pinder, (1996) “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.

21) Lin S-C, Goodman E.D., Punch W.F., (1997) “Investigating parallel genetic algorithms on job shop scheduling problems”, *Evolutionary Programming VI, Proc. Sixth Internat. Conf., EP97, Springer Verlag, NY, P. J. Angeline, et al., eds., Indianapolis,* pp.383-394.

22) Liu, Y., B. S. Minsker, and F. Saied, (2001) "A one-way spatial multiscale method for optimal bioremediation design." *Journal of Water Resources and Planning Management*, 127(2), 130-139.

23) Liu, Y., and B. Minsker (2002) "Efficient multiscale methods for optimal in situ bioremediation design." *Journal of Water Resources and Planning Management*, 128(3), 227-236.

24) Liu, Y., and B. S. Minsker, (2004) "Full multiscale approach for optimal control of in-situ bioremediation." *Journal of Water Resources and Planning Management*, 130(1), 26-32.

25) Maskey, S., Jonoski, A., and Solomatine, D. (2002). "Groundwater remediation strategy using global optimization algorithms." *Journal of Water Resources Planning and Management*, 128(6), 431-440.

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

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

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

29) Minsker, B., Y. Zhang, R. Greenwald, R. Peralta, C. Zheng, K. Harre, D. Becker, L. Yeh, and K.Yager (2003). *Final Technical Report for Application of Flow and Transport Optimization Codes to Groundwater Pump and Treat Systems*, Environmental

Security Technology Certification Program. Available at http://www.frtr.gov/estcp.

30) Punch, W., Averill, R. C., Goodman, E. D., Lin S-C., Ding, Y. (1995), "Design using genetic algorithms - some results for laminated composite structures," IEEE Expert, vol 10 (1), pg:42-49.

31) Reed, P. M., B. S. Minsker, and A. J. Valocchi. (2000) "Cost effective long-term groundwater monitoring design using a genetic algorithm and global mass interpolation." *Water Resources Research*, 36(12), 3731-3741.

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

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

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

35) Tanese R., (1987) “Parallel genetic algorithm for a hypercube”, In Grefenstette J. J., Ed., *Proceedings of the Second International Conference on Genetic Algorithms,* p. 177–183.

36) Wang, C., and Zheng, C. (1998). “Groundwater management optimization using genetic algorithms and simulated annealing: formulation and comparison.” *Journal American Water Resources,* 34(3), 519-530.

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

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