# A SELF-ADAPTIVE HYBRID GENETIC ALGORIHM FOR OPTIMAL GROUNDWATER REMEDIATION DESIGN

BY

FELIPE PATRICIO ESPINOZA

Licen., University of Chile, 1989
M.S., University of Illinois, 2000

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Environmental Engineering in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

# Abstract

Groundwater contamination is the result of multiple human activities, such as agriculture, industrial practices and military operations. The traditional remediation approach is to combine pump-and-treat for plume containment and contaminant capture, with other remediation technology for source control. Pump-and-treat systems are expensive, typically requiring high installation and operation costs. The traditional solution approach is then to proceed by trial-and-error, evaluating different design alternatives and selecting the best one from those evaluated. A large body of research has demonstrated that coupling optimization models with simulation models can aid in identifying effective remediation designs. The simple genetic algorithm (SGA) is a heuristic technique capable of solving these types of problems. Unfortunately, the solution of these complex problems is generally computationally intensive.

This research focuses on the development and use of a hybrid genetic algorithm (HGA), a method that combines the use of SGA with local search to solve a groundwater remediation problem. The inclusion of local search helps to speed up the solution process and to make the solution technique more robust. The result of this research is a highly

reliable numerical tool, the enhanced self-adaptive hybrid genetic algorithm (e-SAHGA) to more efficiently and effectively solve problems using simple genetic algorithms (SGAs). With this tool, the designer can evaluate different solution alternatives in a more timely fashion. A step-by-step methodology has also been developed for evaluating the optimal parameters for using the algorithm. This methodology, together with the adaptive nature of the algorithm, reduces the need for trial-and-error experiments to determine the optimal set of parameters for the algorithm.

The application of the e-SAHGA algorithm to a hypothetical groundwater remediation design problem showed 90% reliability in identifying the solution faster than the SGA, with average savings of 64% across 100 runs with different random initial populations. Finally, e-SAHGA was tested on a field-scale remediation design problem, re-evaluation of the remediation system for Umatilla Army Depot, where it gave computational savings between 30% and 60% and, for one solution method, found a solution that was 4% better than the one found by the SGA.

*To my Mom and in loving memory of my Dad*

# ACKNOWLEDGMENTS

First and foremost I would like to express my gratitude to my advisor Professor Barbara Minsker for being a true mentor, encouraging me in the moments of failure, celebrating the successes, and guiding me during these years, especially during the final steps of this dissertation.

I would like to thank each of my doctoral committee members, Professor Weyland Eheart, Professor David Goldberg, and Professor Albert Valocchi. Professor Valocchi helped me as my adviser during my Master's days allowing me to start this incredible journey. At the same time, Professor Goldberg's insights in genetic and evolutionary computation was key to the final success of my work. Finally, Professor Eheart helped me not only as a part of my doctoral committee but also as my instructor in two very important subjects for my present and future.

I want to thank the entire faculty of Environmental Engineering and Science program for always being accessible and available to me and my fellow graduate students. Specifically, I would like to thank Professors Benito Mariñas and Vernon Snoeyink. Dr. Mariñas helped me to survive my first semester as an international student sharing with me his own experience. Dr. Snoeyink helped me, and other international

students, to improve presentation skills and communications abilities by establishing the CASE study group.

I would like to acknowledge to my fellow group-members at the EMSA research group. I shared with them the experience of creating new science. I also shared with them the pain related with research failure, especially when everything fails and nobody knows why. I want to say thank you in particular to Dr. Patrick Reed who helped me to polish the basic aspects of my research by encouraging me to look for alternative ways to solve my problem.

Last but not least, I would like to express my gratitude to my entire family for their continued support and encouragement from the distance.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

## INTRODUCTION

Groundwater contamination is the result of multiple human activities, such as agriculture, industrial practices and military operations. In the case of agricultural practices, most of the contaminants are pesticides applied to vegetables to prevent diseases. In the case of industrial practices and military operations, most of the contaminants are hydrocarbons and solvents. The National Research Council (1994) estimated that in the United States there are approximately 400,000 contaminated sites. The cost of cleaning these contaminated sites has been estimated at up to $ 1 trillion (National Research Council, 1994).

The traditional remediation approach is to combine pump-and-treat for plume containment and contaminant capture with other remediation technologies for source control (EPA, 1997). A number of enhanced pump-and-treat options are also being used more widely, such as in situ bioremediation and air sparging. Pump-and-treat systems are expensive, typically requiring high installation and operation costs, so finding the least expensive design to achieve cleanup goals is important. At many sites, numerical models are developed to predict the fate and transport of contaminants and assist with remediation design. Using these models, the traditional solution approach is then to proceed by trial-and-error, evaluating different design alternatives and selecting the best one from those evaluated. However, this process is unlikely to find the best design(s) because of the huge number of possibilities involved: multiple well locations, pumping rates (single-stage or constant in time, multi-stage, or variable in time), and types of wells (injection or extraction). For example, the small-scale remediation design test problem presented

in Chapter 4 and used to evaluate the solution methodology proposed in Chapter 3 has approximately 3.9 trillion possible solutions. In fact, if we take into consideration only the designs with 1 well, the number of possibilities is approximately 60,000. Large-scale field sites, such as the one examined in Chapter 6, have even larger numbers of possible solutions.

A large body of research has demonstrated that coupling optimization models with simulation models can aid in identifying effective remediation design (see Chapter 2 for a review of this research). A variety of analytical and heuristic solution alternatives have been proposed in the past to solve this problem. Among these alternatives, this dissertation will focus on genetic algorithms (GAs), which are heuristic techniques based on the biological concept of survival of the fittest. This technique has been proven as or more effective at identifying high quality solutions and it does not require continuity of the objective function or other restrictions such as convexity. The major disadvantage of this technique is computational time. GAs need a relatively large amount of time to search the decision space before the optimal solution is finally attained.

## 1.1 Objectives and Scope

The objective of this research is to develop methods to reduce the computational burden of solving remediation design problems using GAs. This objective will be achieved by combining the GA (which performs global search) with a local search algorithm, creating a hybrid genetic algorithm (HGA). Previous research has shown that HGAs can be more effective than GAs alone and are definitely more effective than analytical techniques (such as gradient-based methods) because they retain the global search capabilities of GAs (see Chapter 2 for a

review of this research). A second important characteristic of GAs is that they do not need to know a priori the location of the feasible decision space in the case of constrained problems such as most groundwater remediation design problems. In fact, for the application of analytical techniques, it is usually necessary to solve the problem multiple times in order to find the best solution, and even then, identification of the global optimum is not guaranteed. Another problem presented by the analytical techniques is the fact that the more robust they are in solving the problem, the longer they take to perform the task, due to time-consuming derivative calculations.

For the reasons previously stated, the goals of this doctoral research are to:

- Design a hybrid genetic algorithm that can be used to solve groundwater remediation problems in a very effective and simple way
- Propose general rules to set GA parameters for optimal application of the HGA
- Apply the HGA to the design of real-world problems that currently cannot be effectively solved because of computational time constraints

## 1.2 Summary of Research Approach

To achieve the objectives outlined above, this thesis addresses a number of issues relevant to the optimal design of an HGA, including concepts of parameter design, population sizing, and optimal use of the computational resources. Chapter 2 summarizes the state of the literature in optimal groundwater remediation design and in the development and application of HGAs both in this field and the Genetic and Evolutionary Computation (GEC) field. Using this background knowledge, Chapter 3 presents a new general purpose self-adaptive hybrid genetic algorithm designed to solve complex problems with continuous variables. Chapter 4 presents the

application of the algorithm to a simple groundwater remediation design problem. Then, Chapter 5 investigates different approaches for improving the algorithm designed in Chapter 3. Finally, Chapter 6 presents the application of the algorithm to the solution of a field-scale, real world groundwater remediation design problem at Umatilla Army Depot. A more detailed summary of the chapters is presented in the following sections.

### *1.2.1 Chapter 2: Literature Review*

Chapter 2 first summarizes the different approaches that have been used to solve groundwater remediation design problems. These approaches include, for example, analytical techniques such as linear programming, non-linear programming, and dynamic programming, or heuristic techniques such as genetic algorithms, tabu search and simulated annealing. The chapter then overviews examples of hybrid genetic algorithms that have been applied in both the water resources field and other fields of study. The chapter also reviews memetic algorithms, which are a type of hybrid genetic algorithm, and a few efforts to develop general purpose HGAs and theory to better understand the characteristics of the HGA. From this chapter we can see how HGAs have been applied to solve a variety of problems ranging from traditional problems in the operations research field to engineering applications (for example water resources, electric utility expansion, and transportation). While numerous applications of HGAs have been proposed, few attempts have been made to create general-purpose HGAs or to examine how theory can guide their effective use. A major goal of this dissertation is to begin addressing this gap.

### *1.2.2 Chapter 3: Development of a New Hybrid Genetic Algorithm*

Chapter 3 presents the development of two general-purpose Hybrid Genetic Algorithms (HGAs): NAHGA, Non-Adaptive HGA, and SAHGA, Self-Adaptive HGA. The chapter includes a complete description of the processes involved in the GA, the linkage between global and local search for both HGAs, the application of local search, and the setting of parameters. The analysis includes a novel way to evaluate the optimal population size for the evaluation of the HGA. The chapter concludes with an evaluation of the HGAs' capabilities by applying both algorithms to traditional test functions from the field of Genetic and Evolutionary Computation. This evaluation shows that both proposed HGAs solve the test problems faster than the SGA for all eight test functions, with savings ranging from 5% to 44%. Of the two algorithms, SAHGA performs better than NAHGA because the adaptive nature of the SAHGA processes allows convergence for a broader set of parameters. Hence, effective use of SAHGA should require little trial-and-error experimentation to identify good parameter values.

### *1.2.3 Chapter 4: Application to a Groundwater Remediation Design Test Problem*

Chapter 4 presents the application of the SAHGA and NAHGA algorithms to a groundwater remediation design problem. The analysis includes the effects of parameter values on performance and the application of the design methodology to evaluate optimal population size and GA parameters. The chapter concludes with an assessment of the relative quality of SAHGA versus NAHGA for the simplest local search algorithm, uniform random search. The results show that both algorithms are capable of solving the problem faster than SGA. The analysis also shows that the behavior of the algorithms for different parameters is usually similar to the behavior presented by the test functions. In some cases, however, the complexities of the

groundwater remediation design problem make the NAHGA algorithm much more sensitive to its parameter values than was seen with the test functions. This shows that the adaptive characteristics of SAHGA are even more beneficial for solving this problem.

### *1.2.4 Chapter 5: Analysis of Algorithmic Enhancements and Local Search Effects on SAHGA Performance*

Chapter 5 tests the SAHGA algorithm on the groundwater remediation test problem from Chapter 4 with more initial populations to test its reliability. In some cases, the algorithm is substantially slower than the SGA. To address this issue, this chapter introduces enhancements to SAHGA including modification of the stopping criterion for the local search algorithm, different approaches to selecting individuals to undergo local search, ending local search when it is no longer effective, and ensuring that the best individual from the local search phase is retained. This chapter also includes evaluation of SAHGA performance for different local search algorithms. Finally, the enhanced algorithm is validated with 100 different initial populations to verify its capabilities in a more extensive test environment. The results of the analysis show that the average savings relative to the SGA improved from 14% to 64% when using the best combination of enhancements. Moreover, the enhanced algorithm achieves 90% reliability in solving the problem faster than the SGA, as compared with the original algorithm that is 80% reliable. It is important to note that the enhancement with the best performance does not include problem-dependent information and can therefore be used to solve other classes of problems.

### *1.2.5 Chapter 6: Application of e-SAHGA Algorithm to Remediation Design at Umatilla Army Depot, Oregon*

Chapter 6 includes the application of SAHGA to a real groundwater remediation problem, the design of the remediation system for Umatilla Army Depot. The site is an abandoned military facility used to store explosives, and it is highly contaminated with TNT and DMX.The problem was solved using two different solution approaches: domain decomposition and full approach. For the first approach, the problem is divided into two sub-problems: well locations and pumping rates. For the first sub-problem, the best well locations are evaluated for constant pumping rates. Then, in the second sub-problem, the optimal pumping rates are evaluated for the best well distribution identified in the first sub-problem. In the second approach, pumping rates and well locations are evaluated simultaneously. For the domain decomposition approach, the savings from applying e-SAHGA are 32.4% compared with SGA; for the full approach the savings are 56.9%. For the full approach, the SGA algorithm also found a solution 4% more expensive than the solution found by e-SAHGA. Overall, the domain decomposition approach was successful in reducing computational effort, resulting in 75.8% savings for the SGA and 61.9% savings for e-SAHGA with no reduction in solution quality.

# Chapter 2

## LITERATURE REVIEW

Chapter 1 introduced the problem of interest: optimal design of groundwater remediation systems by means of hybrid genetic algorithms (HGAs). This chapter provides a literature review on this topic in two major areas: different approaches to solve the optimization problem (Section 2.1) and previous applications and theory related to HGAs (Section 2.2).

### 2.1 Optimization Techniques for Groundwater Remediation Design

Optimization techniques have been applied primarily to traditional pump-and-treat remediation design. These techniques can be divided into two major groups: analytical and heuristic. Among the analytical techniques are linear programming (Aguado et al., 1974; Morgan et al., 1993), nonlinear programming (Gorelick et al., 1984; Ahfeld et al., 1988; and Bear and Sun, 1998; McKinney and Lin, 1996), quadratic programming (Lefkoff and Gorelick, 1986), differential dynamic programming (Chang et al., 1992; Culver and Shoemaker, 1992, 1993, 1997; Whiffen, 1995; Mansfield et al., 1998; and Minsker and Shoemaker, 1998a, 1998b), outer approximation (Karatzas and Pinder, 1993), mixed-integer linear programming (Sawyer and Ahlfeld, 1992; Sawyer et al., 1995; and Gailey 1999), mixed-integer nonlinear programming (McKinney and Lin, 1995), cutting plane (Karatzas and Pinder, 1996), and multiscale combined with differential dynamic programming (Liu, 2001; Liu et al., 2001; and Liu and Minsker, 2002). Examples of heuristic techniques are simulated annealing (Dougherty and Marryott, 1991; Marryott et al., 1993; Rizzo and Daugherty, 1996; and Wang and Zheng, 1998), clustering and random search methods (Maskey et al., 2002), and genetic algorithms (McKinney and Lin, 1994;

Ritzel at al., 1994; Aly and Peralta, 1999a, 1999b; Wang and Zheng, 1998; Smalley et al., 2000; Chan Hilton and Culver, 2000, 2001; Maskey et al., 2002; Erikson et al., 2002; and Gopalakrishnan et al., 2003). A third category of solution approaches is the so-called hybrid methods. In these methods, two analytical and/or heuristic techniques are combined together to solve the problem, For example, Zheng and Wang (1999a) combined tabu search with linear programming, Yu et al. (1998) combined simulated annealing and cutting plane, and Hsiao and Chang (2002) combined genetic algorithms with constrained differential dynamic programming.

One important difference among these different solution techniques is that for non-convex problems most of the analytical techniques are local methods, meaning that the optimal solution found is directly related to the starting point of the search process, and is not guaranteed to be the global optimum.

## 2.2 Hybrid Genetic Algorithms

Hybrid genetic algorithms (HGAs) are a class of hybrid algorithms combining genetic algorithms (GAs) with some type of local search. There is no restriction on the type of local search; it can be an analytical technique, a general heuristic approach (such as random walk), or a problem-specific heuristic designed to exploit the particular characteristics of the problem in study.

Genetic algorithms (GAs) were introduced by Holland in the 1960's and the first important theoretical analysis was presented in the publication *Adaptation in Natural and Artificial Systems* (Holland 1975). This technique is a global search heuristic capable of

searching the entire decision space to find an optimal or near-optimal solution to a particular problem. The approach creates a population of candidate solutions ("individuals") that are then modified through genetic operations (usually selection, crossover, and mutation) in each "generation" to create a new population. With each generation, the population becomes better and better (more highly "fit") until the optimal solution takes over the population.

The second component of the HGA, the local search operator, looks for the best local solution starting at a previously selected point, in this case a solution in the SGA population. From the new point, the algorithm looks again for the best local solution. The algorithm ends when no new point can be found (this is equivalent to a gradient of zero). For functions with multiple local optima, the method finds one local optimum but is not guaranteed to find the global minimum. Local search operators can be classified into two major classes: gradient and non-gradient based methods. Gradient-based methods include steepest descent, Newton's method, conjugate gradient, and many other variations of Newton's method. Examples of non-gradient local search operators are random walk, evolution strategy, and simulated annealing.

In order to understand how HGAs have been applied to solve real problems, Section 2.2.1 presents some examples of application-specific HGAs. Then, Section 2.2.2 presents examples of a class of HGAs called Memetic Algorithms. Finally, Section 2.2.3 presents examples of general HGAs and theory.

### *2.2.1 Application-Specific HGAs*

HGAs have been applied to a variety of problems in different fields and have usually been custom tailored to the particular application. The primary difference among the applications is related to the way local search is used: sometimes it is applied to all of the individuals and sometimes it is applied only to some of them; sometimes it is applied after every generation and sometimes after the last generation only. The following paragraphs present a few representative examples of these applications. A more complete survey on hybrid genetic algorithms can be found in Sinha and Goldberg (2001).

For example, Lin et al. (1995) applied HGA to solve the "traveling salesman problem". In this application, the local search operator was the gradient-based Newton-Raphson method. Local search was applied to all of the individuals in the population and embedded in the GA process. This means that the individuals were created by means of crossover and mutation and modified by local search at the same time. The performance of the algorithm was evaluated in terms of solution quality and not time necessary to achieve that quality.

A second example of HGA is presented by Taguchi et al. (1998). They combined simulated annealing with GAs to evaluate reliability of industrial components. Local search was applied to all of the individuals generated by the GA process in every generation. The results showed that the HGA outperforms the SGA and an improved GA, achieving the same solution in less time.

Another HGA application was presented by Park et al. (1998), in which dynamic programming was applied together with GAs to evaluate "generation expansion plans" for electric utilities. The plans obtained with the HGA outperformed those obtained with the SGA by more than $130 million, but the execution time increased approximately 30%. Local search was applied to all of the individuals generated by the GA in order to improve the quality of the solution, but the improved solutions were not returned to the original GA population. In this way, there was no real interaction between the global and local search.

In transportation engineering, Gen et al. (1998) solved the so-called "bicriteria transportation problem" by combining GAs with linear programming for all of the individuals. Their results showed that the HGA performance was directly linked to the complexity of the problem. In fact, for one problem size, the HGA took more than 3 times the execution time to achieve the same solution, but for a more complex configuration, the HGA execution time decreased more than 60%.

Sabatini (2000) presented an HGA to evaluate optimal time scales for signal analysis. In this application, the local search selected was the gradient-based Newton-Raphson method. In this case, the best individual generated by the GA after the final generation was submitted to the local search algorithm for further refinement. The HGA outperformed the SGA in solution quality with little extra computational effort.

Foster and Dulikravitch (1997) presented an HGA for aerodynamic design. The Nelder-Mead simplex algorithm was applied when the best individual was not improved from one GA

generation to the next. The proposed HGA was able to solve problems that traditional gradient algorithms were unable to solve properly.

For the design of optical transport networks, Sinclair (1999) proposed an HGA where local search was selected from a group of four problem-specific heuristics in every generation based on their previous performance solving the problem. The approach showed good results solving the proposed problems, outperforming the previous solution approaches most of the time.

In the biochemistry field, Nguyen et al. (2002) proposed an HGA for multiple protein sequence alignment. In this algorithm, local search was implemented as a series of specialized heuristics acting in the mutation step of the search and in the generation of the initial population. With respect to quality, their HGA was better than previous approaches for more than 80% of the test problems. The computational effort for the algorithm was similar for simple problems, but for hard problems the HGA outperformed traditional approaches in computer effort and quality.

Mathias et al. (1994) proposed an HGA for seismic data imaging that combined the GA with steepest ascent. In this algorithm, local search was applied to all of the individuals in the population every 10 generations. The application of this algorithm produced solutions that were 7% better and 3 hours faster than previous solution approaches.

For frequency assignment problems, Alabau et al. (2002) proposed an algorithm combining tabu search as a specialized mutation operator during the GA process. The new algorithm outperforms previous approaches for similar computer effort.

Magyar et al. (2000) proposed an adaptive HGA to solve the graph problem called "three-matching problem". In this case, they applied 5 different crossover operators (specific for this type of problem) combined with 3 different gradient-based operators (local search) to evolve the population, together with mutation and selection. The switch between the local search and global search operators is based on a reward mechanism (how successful the operator was in the past). In this way, from the 8 different operators, those that were more successful were called more frequently. For this algorithm, the results are evaluated in terms of solution quality with respect to the best solution available. In this case, the best solution attained by the adaptive GA was the same as the best solution available. No information was given about computational time.

In the water resources field, Cai et al. (2001) solved a reservoir operation model using GA together with linear programming. Once again, the computational performance of the HGA was related to the complexity of the problem, but in every case the HGA outperformed nonlinear programming in terms of solution quality. A second example of an HGA in this field was provided by Kapelan et al. (2002), who calibrated water distribution systems models. They applied one or two iterations of the LM algorithm (similar to Gauss-Newton) to a sample of two or three individuals every ten generations. In their analysis, the HGA gave the same solution quality as the GA, but the algorithm obtained superior performance in terms of computational effort.

Hsiao and Chang (2002) solved a groundwater management problem by uncoupling the problem into two steps. In the first step, the GA evaluated different configurations of pumping well locations and in the second step the fitness of each configuration was evaluated using

constrained differential dynamic programming (CDDP) to identify the optimal pumping rates for the configuration.

### *2.2.2 Memetic Algorithms*

Several researchers have used a special class of hybrid GAs called Memetic Algorithms (MAs) (Moscato 1989; Moscato and Cotta 1999). In this class of HGAs, the information generated by the local search step is exchanged throughout the population after each local search. MAs are based on the concept of cultural evolution, where the "meme" (equivalent to gene) (Dawkins, 1976), the basic unit in the MA, can change over the lifetime of the individual by means of exchange of ideas. In contrast, GAs are based on biological evolution where the "genes" cannot change over the same period of time. This concept of cultural evolution is modeled as local search following some (or all) of the GA processes together with interaction among the individuals during this local search process (exchange of ideas). These local search operators are tailored to fit the particular characteristics of the problem to be solved. The next paragraph presents a few examples of MAs.

Eusuff and Lansey (2003) presented an MA for water distribution network design. In this algorithm, local search is performed by the so-called "shuffled frog leaping" algorithm, and the individuals are grouped in clusters call "memeplexes". In this algorithm, information is exchanged among the individuals in the "memeplex" and among the different "memeplexes", exchanging in effect information throughout the population. This HGA was applied to evaluate the Hanoi water network and the New York city water supply tunnel. In both cases, the HGA outperformed previous approaches in both time and quality. For the quadratic assignment

problem, Merz and Freisleben (2000) proposed an MA with the inclusion of local search after every one of the GA processes of crossover, mutation, and selection. The algorithm finds optimal or near-optimal solutions for the problems tested in shorter time than previous applications. Schönberger et al. (in press) presented an MA designed to evaluate optimal timetables for non-commercial sport leagues. The most important component of this application is the high number of constraints that must be satisfied in order to have a successful timetable. For this reason, the local search part of the problem was designed to reduce the number of constraint violations. The algorithm performs reassignments and then is fed back to the population. The algorithm found good quality solutions with reduced execution time.

### *2.2.3 General HGA Algorithms and Theory*

The previous HGA algorithms were generally developed and tailored to specific applications. A few researchers have developed more general HGAs that can be used for different types of problems or for better understanding of the HGA processes. For example, Tang et al. (1998) proposed an HGA to solve non-linear problems that combined gradient search directly into the mutation operator to solve constrained problems without using a penalty for infeasible solutions. This new approach outperformed the more traditional approach of including penalty functions to solve constrained problems. Kazarlis et al. (2001) proposed an HGA combining GAs with micro GAs (MGA) as the local search algorithm. MGA operates over very small populations for a few generations to find the best solution in the neighborhood of the starting solution.

In another study of HGAs, Lobo and Goldberg (1997) presented a decision-making approach to design of HGAs. In this work, they studied when to move from global search to local search and how to use the local and global search in the best way possible. Later, Goldberg and Voessner (1999) and Sinha and Goldberg (2001) presented an analytical approach to optimizing the execution time between local and global search (using a random search method, not GA) in order to achieve an acceptable reliability level.

Hart (1994) presented an evaluation of the interaction between GAs and local search algorithms using both analytical and heuristic approaches. He also included the concept of adaptive or non-adaptive local search. In his framework, a local search algorithm is adaptive when the selection of the individuals to undergo local search is performed taking into consideration the individuals selected previously. On the other hand, non-adaptive algorithms are those where the selection of the individuals is performed independently in every iteration. For the adaptive algorithms, the adaptation is performed using the fitness information for each individual to avoid selection of redundant individuals (individuals with the same or nearly the same fitness). He used this algorithm to solve several applications, including training neural networks, molecular conformation, and drug docking (molecules design). The results of these applications showed that the HGA (with heuristic local search) outperformed other solution techniques such as GAs or Monte Carlo simulation. His results also showed that the inclusion of powerful local search algorithms (such as gradient-based algorithms) did not improve the performance of the HGA, and in fact sometimes gave worse performance than the GA alone.

# Chapter 3

## DEVELOPMENT OF A NEW HYBRID GENETIC ALGORITHM

Chapter 1 illustrated the problem of interest, groundwater remediation design, and the tool selected to solve it, the hybrid genetic algorithm (HGA). HGAs combine GAs and local search to reduce the computer burden of solving problems with GAs. Then, Chapter 2 presented how HGAs have been applied in different fields. That chapter also shows the limitations of HGAs, such as the problem-specific nature of the algorithm and the static nature of the processes involved. It is desirable to have an HGA capable of solving complex problems without using the specific characteristics of the problem, allowing generalization to different types of problems. With this objective in mind, this chapter presents two hybrid genetic algorithms (NAHGA and SAHGA) that can be used to competently solve optimization problems. First, each algorithm is described and a methodology is presented for selecting the HGA algorithms' parameters and population size for optimal performance in Section 3.1. Then Section 3.2 presents a set of traditional functions to evaluate the performance of the proposed algorithms. Section 3.3 describes the five different local search algorithms to be used in this analysis and Section 3.4 performs population sizing for each algorithm and test function. Finally, the HGAs' performance is compared with the Simple Genetic Algorithm (SGA) on the test problems in Section 3.5. Conclusions and recommendations are given in Section 3.6.

### 3.1 Hybrid Genetic Algorithm Development

This section describes the HGA components: local search, global search using the genetic algorithm, and linkage of the two components by means of evolution. The rationale behind the

design of the HGAs is presented first, along with the basic elements of the HGA, followed by detailed descriptions of the two HGAs used in this work (NAHGA and SAHGA). Finally, a methodology for selecting HGA parameters is described that ensures effective use of the algorithm.

### *3.1.1 Basic Elements*

In general, an HGA consists of a simple genetic algorithm combined with a local search algorithm. Fig. 3.1 gives a block diagram of the algorithm that shows the interaction between local and global search (SGA). The first element in the figure is the creation of the initial random population and fitness evaluation as in the SGA. The SGA used in this work is defined by four basic operators (Goldberg, 1989; Schwefel, 1975): binary tournament selection, uniform crossover, simple mutation, and $\mu+\lambda$ selection, which are discussed below. The first step in the SGA is to create an initial population of possible solutions. Each solution is written in binary form and stored in a "chromosome" that represents the decision variable values for that solution. The next step is the application of the SGA in a sequential process: reproduction (creation of a child population) and then selection (from parent and child populations). First, during the binary selection process, two individuals of the parent population are randomly selected and the best one is selected to be parent 1. Then the process is repeated to select parent 2. Second, uniform crossover is applied to the two individuals previously selected to create two new individuals. During crossover, the genetic information between the individuals is exchanged gene by gene with probability $P_c$. From these two individuals, the best one is selected for the child population. The next step is the application of mutation, in which bits from individuals in the child population are changed from "1" to "0" or vice versa with probability $P_m$. The final step is the application of the $\mu+\lambda$ selection scheme, where $\mu$ represents the parent population and $\lambda$ the

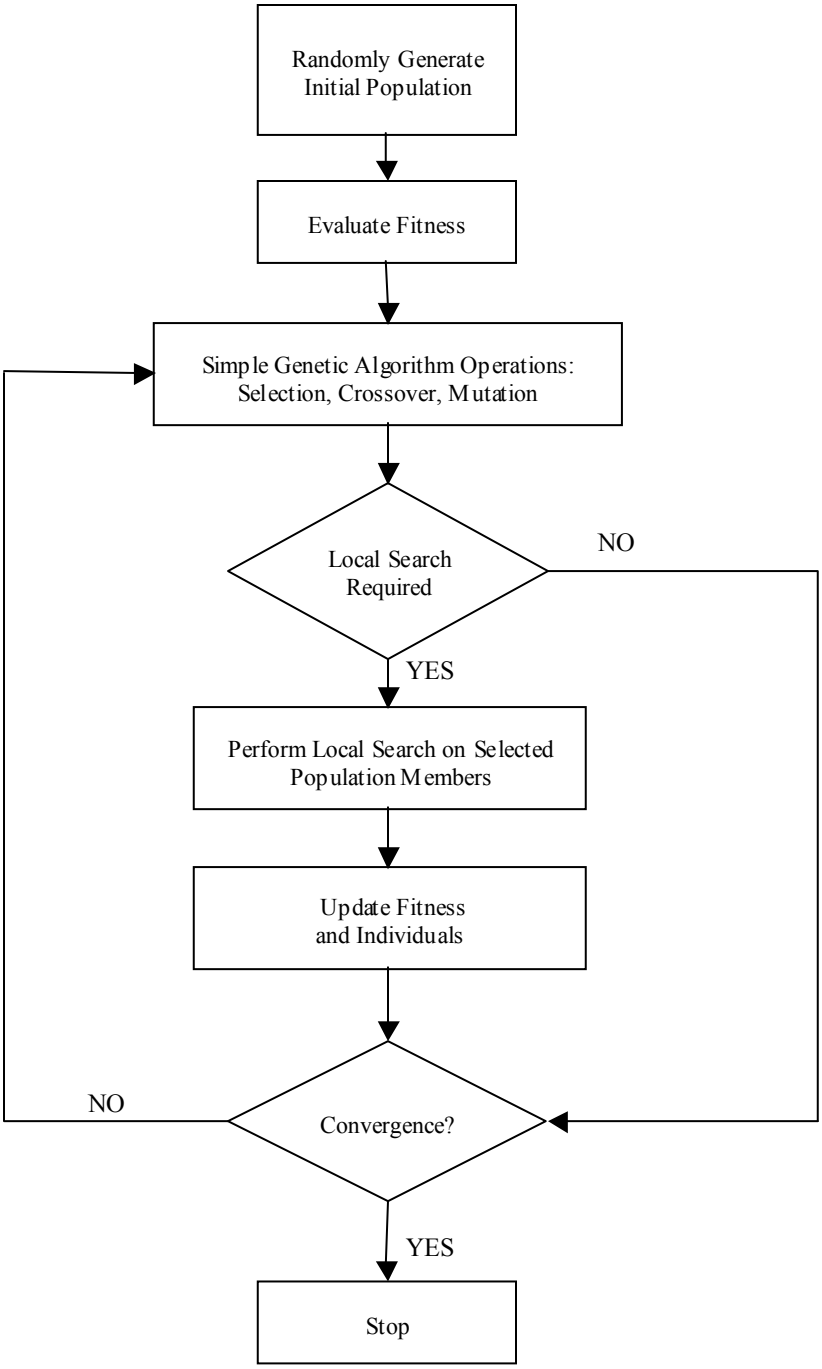



**Fig. 3.1 Block diagram for the HGA**

child population. In this selection mechanism, the next generation is created from the $\mu$ best individuals in the total population of $\mu+\lambda$ individuals. Preliminary trials showed that this approach performed better than binary tournament selection alone. Through the successive application of these operators, the initial population of solutions is evolved into a highly fit population, where fitness is defined by the objective function and constraints of the problem. Then the necessity of local search is evaluated and, if necessary, local search is applied to a small number of individuals in the population. In the next step, the results of the local search are used to update the individuals.

To combine the SGA and local search, HGAs typically use one of two approaches: Lamarckian or Baldwinian evolution (Hinton and Nolan, 1987; Whitley et al., 1994). Lamarck (1802) presented his theory of learned evolution, in which direct learning passes the best characteristics of each individual from generation to generation. This means that both the change in genotypic information and fitness are passed to the individual as genotypic information at the end of the local search step (i.e., the chromosome of the individual is changed). Baldwinian evolution, also known as the Baldwin effect (Baldwin, 1896), is survival of the fittest following the direction of learning. In this case, only the improved fitness function value is changed after local search and not the genotypic information. Lamarckian evolution has been shown to cause faster convergence than Baldwinian evolution, but sometimes causes premature convergence (Whitley et al., 1994). To test the effects of Lamarckian versus Baldwinian evolution, an HGA parameter PB is defined to be the proportion of individuals undergoing Baldwinian evolution out of the total number of individuals undergoing local search.

### *3.1.2 Non-Adaptive Hybrid Genetic Algorithm (NAHGA)*

The NAHGA algorithm is a standard, non-adaptive hybrid genetic algorithm that combines an SGA with local search. The local search step is defined by three basic parameters: local search frequency, probability of local search, and number of local search iterations. The first element for the definition of the algorithm is the local search frequency, which is the switch between global and local search. In the NAHGA algorithm, this switch is performed every $\Delta G$ global search generations, where $\Delta G$ is a constant number called the generation gap. For example, if $\Delta G=3$, local search would be performed every 3 generations during the SGA. The second element of the algorithm is the probability of local search (P), which is the probability that local search will be performed on each member of the SGA population in each generation where local search is invoked. This probability is constant and is defined before the application of the algorithm. Finally, each time a local search is performed, it performs a constant number of local search iterations (LS) before the local search is halted.

Note that it would be possible to design the NAHGA with fewer parameters. For example, the parameters for local search frequency and the probability of local search could be combined into a single parameter, the probability of local search. For example, performing local search on 10% of the population every 2 generations is equivalent to performing local search on 5% of the population every generation (from the point of view of function evaluations). While the number of parameters required to be tuned is reduced under this approach, it does not allow adaptation of the algorithm to recent performance, an important feature of SAHGA (see below). To allow more direct comparisons of the two HGA algorithms, then, the two parameters are separated into P and $\Delta G$ in this work.

### *3.1.3 Self-Adaptive Hybrid Genetic Algorithm (SAHGA)*

The SAHGA algorithm works with the same operators as the NAHGA algorithm: local search frequency, probability of local search and number of local search iterations. The major difference in the approaches is that the SAHGA adapts in response to recent performance of the algorithm as it converges to the solution. The details of the adaptations are given below.

**a) Local Search Frequency.** Instead of a constant generation gap, local search is invoked only when the SGA's performance indicates that it is needed. In order to select the optimal time for local search we need to understand how a GA finds the optimal solution. In the generational process, the fitness starts as a random distribution with mean and standard deviation unknown and converges to a uniform distribution with mean equal to the solution of the problem and approximately zero standard deviation. We propose tracking changes in performance that indicate a need for local search by evaluating changes in the mean and standard deviation of the fitness. The most obvious choice of performance measures would be to track one of the following two ratios: the ratio of mean fitness in generation $i$ to mean fitness in generation $i$-1 or the ratio of the standard deviation of fitness in generation $i$ to $i$-1. These two ratios are shown in Fig. 3.2 for the SGA alone, solving for one of the test problems presented in Section 3.2 (Griewank). From this plot, it is clear that both the mean and the standard deviation of fitness change substantially during evolution. Therefore, the best way to track the change in performance is to track the combined change in mean and standard deviation using the coefficient of variation, which is defined as the ratio of the standard deviation and the mean of the population fitness. Fig. 3.3 shows the coefficient of variation (CV) for the same experiment.

The overall trend for the coefficient of variation is decreasing and approaching to zero as the population converges to the optimal solution.

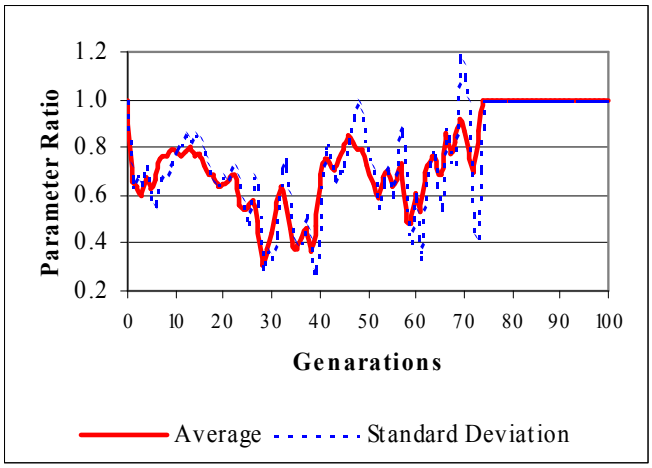



**Fig. 3.2 Average and standard deviation ratio**

To track changes in the coefficient of variation, we need a new parameter (CVR), the coefficient of variation ratio, given in Eq. (3.1) and shown in Fig. 3.3 (dashed line):

$$CVR = \frac{CV(i)}{CV(i-1)} \tag{3.1}$$

where CV(i) is the coefficient of variation at generation "i." CVR represents the relative change in the coefficient of variation from one generation to the next. When CVR >1, the solution at generation "i" is worse than the solution at generation "i-1", usually because the algorithm is exploring a new area of the decision space. When a new area of the decision space is discovered through a crossover or mutation operation, the fitness of the new solution may be quite different from the rest of the population, causing an increase in the standard deviation of the fitness or a decrease in the mean fitness, either of which would increase the coefficient of variation and cause CVR to be greater than one. Thus, when CVR>1, one or more promising new solutions

have been identified and local search may be beneficial for further exploring the new area of decision space.

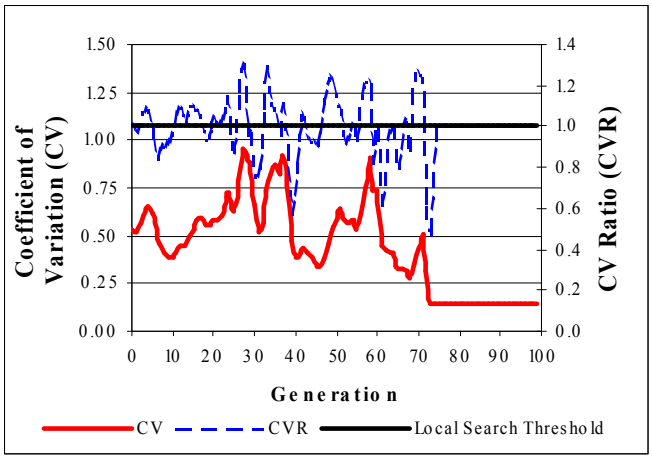



**Fig. 3.3 Global search-local search threshold effect**

In the example shown in Fig. 3.3, CVR values greater than one were found to be good indicators of promising solutions that could benefit from local search to improve the solution of the problem. Given these strong correlations, the local search frequency in the SAHGA was defined using a threshold (T), where local search is invoked whenever CVR > T. In Fig. 3.3, a threshold (T) of one is shown on the CVR curve to illustrate when local search would be invoked if T were set to one.

**b) Probability of Local Search Selection.** In an HGA, local search typically operates over a small portion of the total population because the additional function evaluations required for local search can be computationally expensive. Therefore, when local search is achieving better performance than the most recent global search iteration (using the criterion shown in the

next section), the SAHGA algorithm is adapted to search a smaller portion of the population using the relationship shown in Eq. (3.2).

$$P = \begin{cases} P_0 & n = 1 \\ P_0\,(1-\varepsilon)^{n-1} & n > 1 \end{cases} \tag{3.2}$$

In this equation, the local search probability P decreases in a constant way from the initial value. $P_0$ is the user-specified initial value of the local search probability, "n" is the local iteration number in the local search step, and "ε" is a user-specified parameter governing the rate of decrease in the local search probability. The probability P is reset to $P_0$ at the beginning of every local search step in order to start with the same sampling size at the beginning of every local search.

**c) Number of Local Search Iterations.** One important issue for the application of the algorithm is how long the local search continues before switching back to the global GA search. In order to make this decision, we compare the most recent fitness improvement by local search with the latest fitness improvement by global search. This criterion is presented in Eq. (3.3):

$$\text{do local search if } \frac{\Delta\text{Global}}{\text{pop}} < \frac{\Delta\text{Local}}{\text{fev}} \tag{3.3}$$

where ΔGlobal is the improvement in the maximum fitness (i.e., the fitness of the best member of the population) achieved in the most recent generation without local search, ΔLocal is the current fitness improvement in the local search step for the individuals undergoing local search, pop is the population size (which is the number of function evaluations required for global search), and fev is the number of function evaluations required for the local search step. This criterion scales the fitness improvements by the computational effort required (pop and fev) so

that the ratios are comparable. When Eq. (3.3) is no longer true, or when the number of iterations exceeds a user-specified maximum value (LS), the algorithm switches back to global search.

## 3.2 Test Functions

In order to evaluate the performance of the algorithm, the following 8 test functions were selected from the GA literature:

- Unimodal problems: De Jong's 1 and De Jong's 2 (De Jong, 1975)
- Multimodal problems with the same local mimimum at different locations: Branin (Branin, 1972) and Six Hump (Dixon and Szego, 1978)
- 2 multimodal problems with different optima: Schwefel (Schwefel, 1975) and Griewank (Bäck et al., 1997)
- 2 constrained unimodal problems: Test08 (Kim & Myung, 1997) and Bracken & McCormick (Bracken and McCormick, 1970)

## 3.3 Local Search Algorithms

The local search operator attempts to find the best solution starting at a previous selected point, in this case a solution in the SGA population. For this analysis, the following five local search algorithms were selected:

- Random Walk with Uniform Distribution (LS1): In general, the random walk is simply the movement from one point of the decision space to a new point randomly selected using a uniform distribution from a neighborhood around the starting point (Spitzer, 1964). One iteration of this algorithm requires one fitness function evaluation.

- Random Walk with Normal Distribution (LS2): This algorithm is similar to the uniform distribution discussed previously, but the change of location is evaluated with a normal distribution instead of a uniform distribution. For this reason, the points located near the starting point are more likely to be selected than those located closer to the boundary of the search area. Again, one iteration requires one function evaluation.
- (1+1)-Evolutionary Strategy (LS3): This algorithm, proposed by Rechenberg (1973) and Schwefel (1981), randomly selects a new location using a normal distribution with variable standard deviation. The standard deviation changes following the so-called $1/5$ success rule. When the probability of success is less than $1/5$, the standard deviation is reduced, and when the probability of success is more than $1/5$, the standard deviation is increased. The probability of success is defined as the ratio between the number of successful searches and the total number of searches. This algorithm also requires one fitness function evaluation per local search iteration.
- Random Derivative (LS4): This algorithm first randomly selects a search direction. Then, the derivative of the fitness function is calculated in this direction. Using this direction and a constant step a new point in the space is selected starting from the original one. This algorithm needs 2 fitness function evaluations for every local search iteration (one for the evaluation of the direction and one for the evaluation of the fitness for the new point).
- Steepest Descent (LS5): The algorithm evaluates the new point following the direction of the gradient of the function at the starting point. This algorithm performs one function evaluation for every one of the decision variables of the particular test problem (to

numerically evaluate the gradient of the fitness function) and one function evaluation to evaluate the fitness of the new individual.

## 3.4 Methodology For Setting HGA Parameters

### *3.4.1 Description*

One of the critical elements for optimal performance of a GA is the population size and parameter setting. The traditional approach is a trial-and-error setting of the parameters and population size, but this approach can require numerous trial-and-error runs. For example, Aly and Peralta (1999a) required sixty trial runs to obtain good performance. In order to overcome this time-consuming trial-and-error process, Reed et al. (2000) proposed a three-step design methodology for parameter settings and population size using theoretical results developed by Thierens (1995), De Jong (1975), and Harik et al. (1997). The recommendations for parameters are:

- For tournament selection of tournament size s, Thierens (1995) showed that the upper bound of the crossover probability, $P_c$, is given by:

$$P_c \leq \frac{s-1}{s}. \tag{3.4}$$

- To ensure maximum innovation, Reed et al. (2000) recommended setting $P_c$ equal to the upper bound. For binary tournament selection, used in this work, s=2 and hence $P_c = 0.5$.
- De Jong (1975) showed that SGA gives good performance when the probability of mutation $P_m$ is set using the relationship:

$$P_m \approx \frac{1}{N} \tag{3.5a}$$

where N is the population size.

An alternative expression to evaluate the probability of mutation is given by:

$$P_m \approx \frac{1}{L} \tag{3.5b}$$

where L is the chromosome length.

In this application, we select the first equation to evaluate probability of mutation because we are working with small population sizes, so it will give a higher mutation rate. We have found a higher mutation rate to give more effective performance.

For the population size evaluation, a relationship derived from the random walk model proposed by Harik et al. (1997) is used. In their approach the population size is given by:

$$N \geq -2^{K-1} \ln(\alpha) \left( \frac{\sigma_{bb} \sqrt{\pi (m-1)}}{d} \right) \tag{3.6}$$

In Eq. (3.6), K represents the building block (BB) order, $\alpha$ is the reliability (i.e., the probability that the GA finds the optimal solution); $\sigma_{bb}$ is the standard deviation of the BB fitness; m is the maximum number of BBs within a single string, and d is the signal difference between the best and second best solution. The application of Eq. (3.6) to evaluate the population size is not feasible because the building blocks are generally unknown. However, Eq. (3.6) can be approximated by (Harik et al., 1997):

$$N \geq -2^{K-1} \ln(\alpha) \left( \frac{\sigma_f}{d} \right) \tag{3.7}$$

where $\sigma_f$ is the standard deviation of the fitness function. The parameters $\sigma_f$ and d are estimated using a large, random initial population. The building block order, K, is unknown but can be assumed to vary between 1 and 5 (Reed at al., 2000). In their design methodology, K starts at 1 and is increased until the difference between optimal solutions from 2 consecutive runs is not significant and the solution has taken over at least 80% of the population.

For the HGA case, Eq. (3.7) is modified to include the effect of local search. During local search, the diversity of the initial population is decreased because multiple members of the population usually have the same nearby local optima. Fig. 3.4 shows the fitness for a random population before and after local search, where $\mu$ is the average population fitness and $\sigma$ is the standard deviation of fitness. In this analysis, all of the individuals in the population undergo local search, which represents an upper bound for local search improvement. From Fig. 3.4, it is clear that the standard deviation of the fitness for the population after local search is substantially reduced. Initially the average is 30 and the standard deviation is 27.1. After 3 iterations (60 function evaluations) uniform random search reduces the standard deviation to 24.6 and the average to 25.2. Steepest descent local search performed 1 iteration (60 function evaluations) and changed the standard deviation to 7.3 and reduced the average to 12.1. Steepest descent local search lowered the standard deviation and the average by an amount larger than uniform random local search while using the same 60 function evaluations. Both methods of local search performed the same number of function evaluations, however the gradient method only did 1 iteration while the uniform random method did 3 iterations. These results show that gradient search decreases diversity faster than random search. This effect can be modeled as $\sigma_{f1} = \beta\,\sigma_f$, where $0 \le \beta < 1$. Using the previous information, Eq. (3.8) can be rewritten as:

$$N \ge -2^{K-1}\,\ln(\alpha)\left(\beta\,\frac{\sigma_f}{d}\right) \tag{3.8}$$

The parameter $\beta$ can be estimated by applying local search to the initial random population for a predefined number of iterations. $\beta$ would then be equal to the ratio of the standard deviations after and before local search ($\sigma/\sigma_0$), where $\sigma_0$ is the initial standard deviation. For example, Fig. 3.5 shows the parameter $\beta$ for the Griewank test function using the

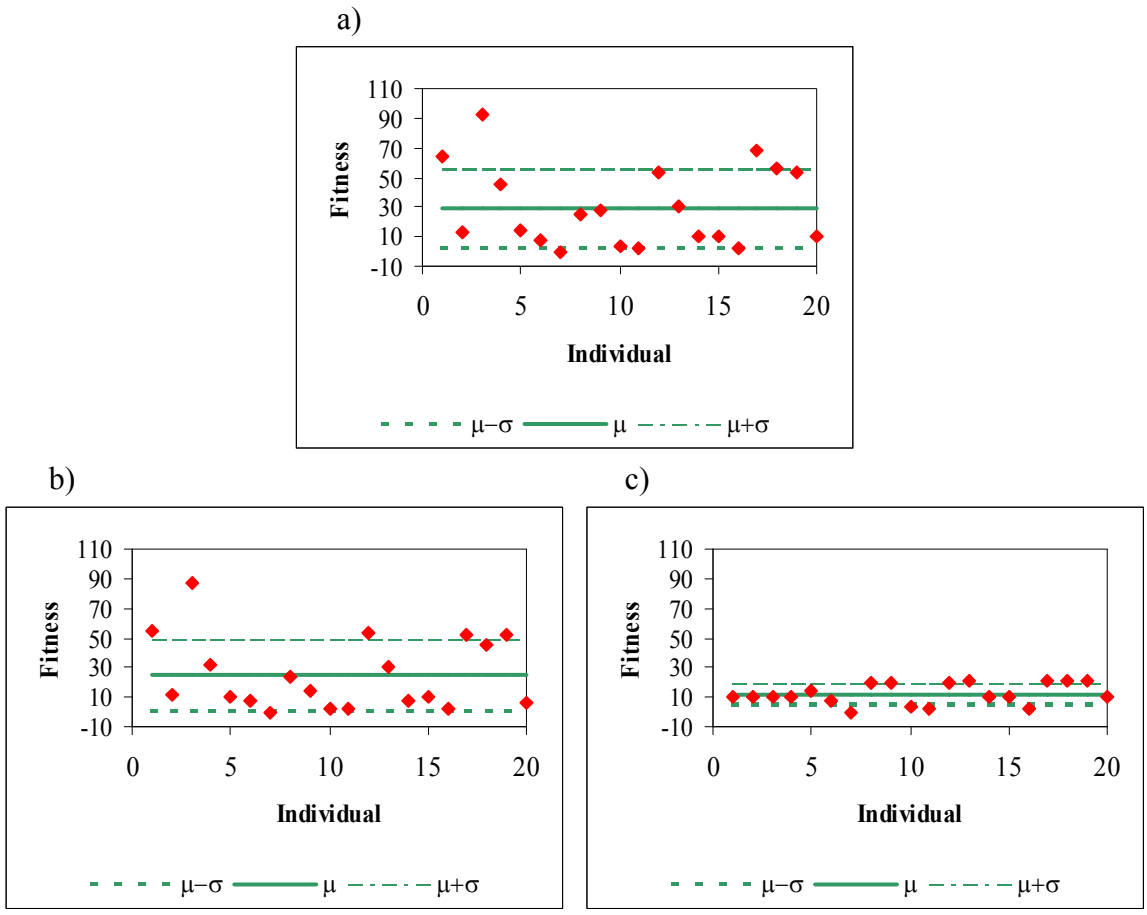



**Fig. 3.4 Local search effect on fitness: a) before local search, b) after random uniform search (LS1), c) after gradient search (LS5)**

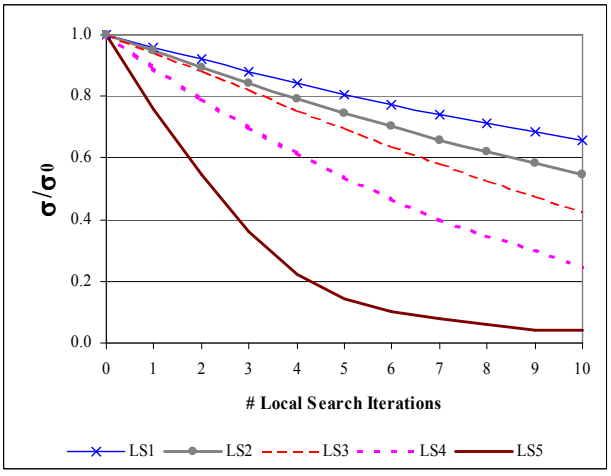



**Fig. 3.5 Standard Deviation reduction for different local search algorithms (Griewank)**

five local search algorithms. From Fig. 3.5, it is clear that gradient (LS5) has the largest effect on the standard deviation, but 1 iteration of this algorithm is equivalent to 3 iterations of LS1, LS2 or LS3 in terms of computational effort. The random derivative (LS4) method is the second most effective local search method at decreasing the standard deviation of the population. The first three local search methods also have substantial effects on the standard deviation, especially if the amount of effort involved is taken into consideration. Fig. 3.5 shows that β changes from one iteration to the next. A reasonable estimate can be obtained by using the average value of β across the maximum number of local search iterations (LS), for this case 5 iterations. In this way, the estimate of β is an average that takes into consideration the effect of different numbers of local search iterations that are likely to occur during a run. Table 3.1 shows the values of β calculated in this manner for each algorithm used in this study.

**Table 3.1 Standard Deviation Reduction**

| Local Search | β |
|---|---|
| LS1 | 0.86 |
| LS2 | 0.82 |
| LS3 | 0.79 |
| LS4 | 0.67 |
| LS5 | 0.36 |

Once β has been estimated, the next step in estimating population size is the evaluation of the parameters $\sigma_f$ and d in Eqs. (3.7) and (3.8). Estimating $\sigma_f$ is a straightforward calculation of the standard deviation of fitness of the same initial random population of 1,000 individuals used for estimating β. Since $\sigma_f$ reduces as the algorithm progresses toward convergence to a single solution, this initial estimate gives a conservative value of population size. Estimating d, the signal difference between the best and the second best members of the population, is more

difficult. In Harik et al. (1997), d is defined as the difference in fitness of the best and second best building block in the population. For real applications, the building blocks are unknown. For this reason, in this application, d is approximated as the differences in fitness of the best members of the population, working with a probabilistic approach based on frequency analysis theory (Hogg and Craig, 1978), in which the histogram of the fitness function is evaluated. This histogram represents different "classes" of fitness that can be statistically identified. Using the histogram, "d" is evaluated as the size of the first class of the histogram. Fig. 3.6 shows the histogram for the initial random population for the Griewank test function.

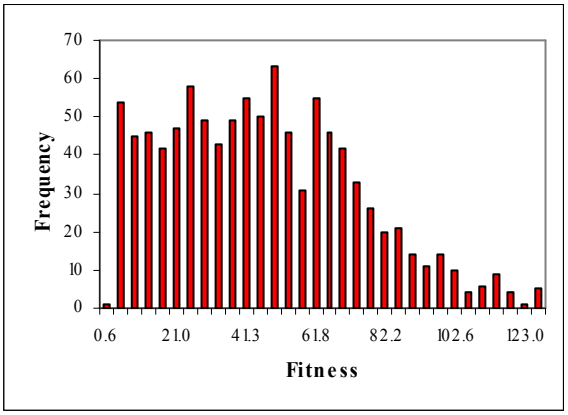



**Fig. 3.6 Histogram for test function Griewank**

Using this information, the initial population for the SGA and the HGA can be estimated from Eqs. (3.7) and (3.8), respectively, for different building block orders. The results of this process for the Griewank test function are presented in Table 3.2. The results show clear differences in population sizes from the different standard deviation reductions (β). The evaluation of the population sizes for each building block order follows the same procedure for the other test functions; the results are given in Appendix A, Table A.1.

**Table 3.2 Population Size for SGA and HGA (Griewank)**

| Building Block Order (K) | SGA | HGA | | | | |
|---|---|---|---|---|---|---|
| | | LS1 | LS2 | LS3 | LS4 | LS5 |
| 1 | 20 | 18 | 17 | 16 | 14 | 8 |
| 2 | 40 | 36 | 34 | 32 | 28 | 16 |
| 3 | 80 | 72 | 68 | 64 | 56 | 32 |
| 4 | 160 | 144 | 136 | 128 | 112 | 64 |
| 5 | 320 | 288 | 272 | 256 | 224 | 128 |

### *3.4.2 Population Size for Test Functions*

The next step in the analysis is the selection of the population size from the possible values for each building block order (given in Table 3.2 and Appendix A). To achieve correct convergence, the convergence time must be less than the drift time, which is $t_{drift} \approx 1.4$ N (where N is the population size) (Thierens et al., 1998). The convergence time can be evaluated using the relation $t \approx 2l$ *(*where $l$ is the chromosome length) (Thierens et al., 1998). Imposing the condition that convergence time must be less than drift time to obtain the correct solution, the population size must satisfy the relation presented in Eq. (3.9), which represents the lower boundary for population size:

$$N > 1.42\, l \tag{3.9}$$

Using Eq. (3.9) and the results presented in Table 3.2 and Appendix A, Table 3.3 shows the adopted SGA population size for all eight test problems, which was chosen to be the smallest K value that exceeded the lower boundary. Table 3.3 also includes the chromosome length, the lower boundary for the population and the minimum building block order.

The final step in the evaluation of HGA population size is to combine the adopted population for the SGA with the standard deviation reduction parameter (β), shown in Table 3.4. Table 3.5 shows the final HGA population adopted for each test problem and local search

algorithm in study. From Table 3.5 it is clear that for some test problems (DJ1 and Six-Hump) the choice of local search algorithm has no effect on the standard deviation reduction, but for others (Schwefel, Griewank, and B&M) the standard deviation reduction varies for different local search algorithms and for this reason the population necessary for the application of the HGA varies as shown in Table 3.5.

**Table 3.3 Adopted SGA Population for Test Problems**

| Function | Chromosome Length | Lower Boundary | Minimum Building Block Order (K) | Adopted Population |
|---|---|---|---|---|
| **DJ1** | 150 | 213 | 5 | 320 |
| **DJ2** | 60 | 83 | 4 | 96 |
| **Branin** | 60 | 83 | 4 | 160 |
| **Six-Hump** | 60 | 83 | 5 | 160 |
| **Schwefel** | 150 | 213 | 4 | 280 |
| **Griewank** | 60 | 83 | 4 | 160 |
| **Test08** | 60 | 83 | 4 | 128 |
| **B&M** | 60 | 83 | 3 | 96 |

**Table 3.4 Standard Deviation Reduction for Test Problems (β)**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 0.90 | 0.90 | 0.90 | 0.90 | 0.90 |
| **DJ2** | 0.67 | 0.67 | 0.67 | 0.42 | 0.42 |
| **Branin** | 0.60 | 0.60 | 0.50 | 0.30 | 0.30 |
| **Six-Hump** | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 |
| **Schwefel** | 0.86 | 0.80 | 0.77 | 0.57 | 0.57 |
| **Griewank** | 0.86 | 0.82 | 0.79 | 0.67 | 036 |
| **Test08** | 0.94 | 0.88 | 0.88 | 0.88 | 0.63 |
| **B&M** | 0.96 | 0.92 | 0.92 | 0.50 | 0.50 |

## 3.5 Evaluation of HGA Performance

This section presents the evaluation of the performance of the algorithm for the test problem Griewank. The results for the other test problems are presented in Appendix A (Fig. A.1

to A.35). The first set of experiments compare the performance of SAHGA for the different parameters involved in the HGA analysis: local search frequency, probability of local search, number of local search iterations, adaptive parameter, and proportion of Baldwinian evolution. The second set of experiments compares the HGA algorithms' performance to the SGA. These experiments were performed for 100 runs with different random seeds, creating different initial populations.The performance of the algorithms was measured using time, in terms of the total number of function evaluations necessary to converge to the optimal solution with an error no greater than 0.01%.

**Table 3.5 HGA Population for Test Problems**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 288 | 288 | 288 | 288 | 288 |
| **DJ2** | 64 | 64 | 64 | 40 | 40 |
| **Branin** | 96 | 96 | 80 | 48 | 48 |
| **Six-Hump** | 96 | 96 | 96 | 96 | 96 |
| **Schwefel** | 240 | 224 | 216 | 160 | 160 |
| **Griewank** | 144 | 136 | 128 | 112 | 64 |
| **Test08** | 120 | 112 | 112 | 112 | 80 |
| **B&M** | 92 | 88 | 88 | 48 | 48 |

### *3.5.1 Parameters*

For the analysis of the performance of the algorithms with respect to the different parameters, a default set of parameters was selected based on previous experience working with test functions (Espinoza et al., 2001). The default set of parameters is:

- Local search frequency: $\Delta G=3$ (NAHGA) and $T=0.75$ (SAHGA)
- Probability of local search: $P=0.1$ (NAHGA) and $P_0=0.1$ (SAHGA)
- Maximum number of local search iterations: LS= 5

- Adaptive parameter for local search probability (SAHGA): ε= 0.1
- Proportion of Baldwinian evolution: PB= 0.25

The effects of each parameter on performance were examined by varying each parameter, leaving the remaining parameters at their default values. More details follow for each parameter.

**a) Local Search Frequency.** The first experiment was designed to evaluate the effects of local search frequency on the solution of the problem. For the NAHGA algorithm, local search was performed at a pre-defined interval ΔG; for the SAHGA algorithm, local search followed the threshold requirements previously explained. Fig. 3.7 a) and b) show the results for the NAHGA and SAHGA algorithms, respectively. For NAHGA, the optimal results are somewhat different for the 5 local search algorithms. For example, for LS5 the best performance is achieved when the local search is applied every 5 generations. On the other hand, for LS1 and LS2 the performance of the algorithm improves when the generation gap increases. For the SAHGA algorithm, the optimal results are almost identical for LS1 to LS4, independent of the value of the parameter T. For LS5, the performance improves somewhat when T increases. For the other test problems, the behavior of the algorithms is mostly similar. However, for the constrained test problems (Test08 and B&M), there are more variations in performance for different local search frequencies for the NAHGA algorithm. (See Appendix A, Fig. A.28 a and Fig. A.33 a).

**b) Local Search Probability and Adaptive Parameter.** The second experiment tested the effect of the probability of local search parameters on performance. In SAHGA, the probability of local search is adapted using the parameter ε in Eq. (3.2). Fig. 3.8 shows the effect

of the adaptive parameter for the default initial probability of local search. This experiment shows that there is only a slight improvement in performance for different values of the adaptive parameter for every local search algorithm. For this function, ε=0 (i.e., no adaptation of the local search probability) performs as well as the settings with adaptation. However, for other functions (e.g., see Fig. A.6 in Appendix A), the adaptation produces better performance. On the other hand, good performance is also achieved for a value of ε=1 for all functions. This means that only 1 local search iteration is performed (see Eq. (3.2)). This result shows that the first iteration of local search is the most essential for these test functions and adaptation may not always be necessary.

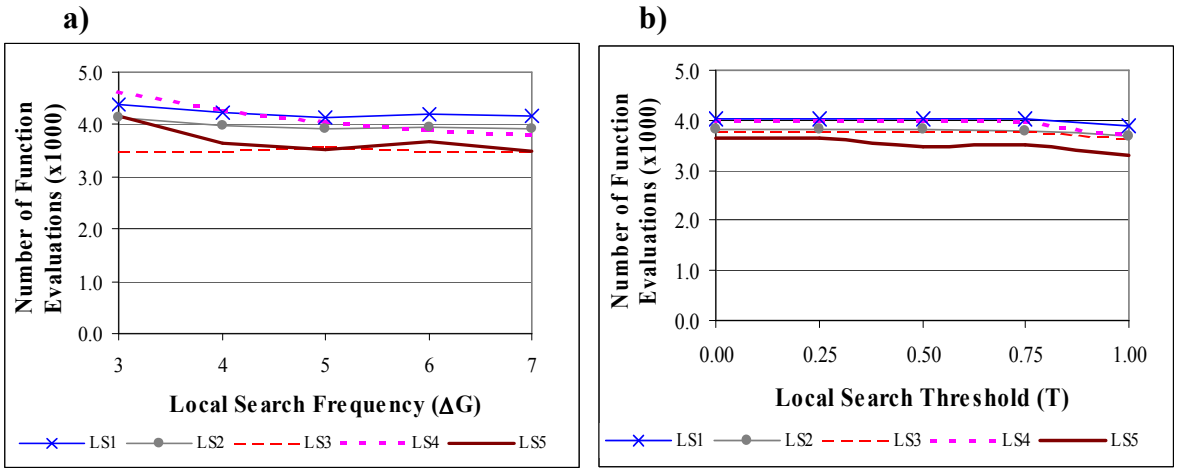



**Fig. 3.7 Effects of varying local search parameters: a) local search frequency for NAHGA and b) local search threshold for SAHGA**

The next parameter to study is the probability of local search, which determines the number of individuals undergoing local search. Fig. 3.9 a) shows the results for NAHGA and Fig. 3.9 b) for SAHGA for the default value of ε. From these results, it is clear that the optimal performance of NAHGA for these default parameter settings is when the probability of local

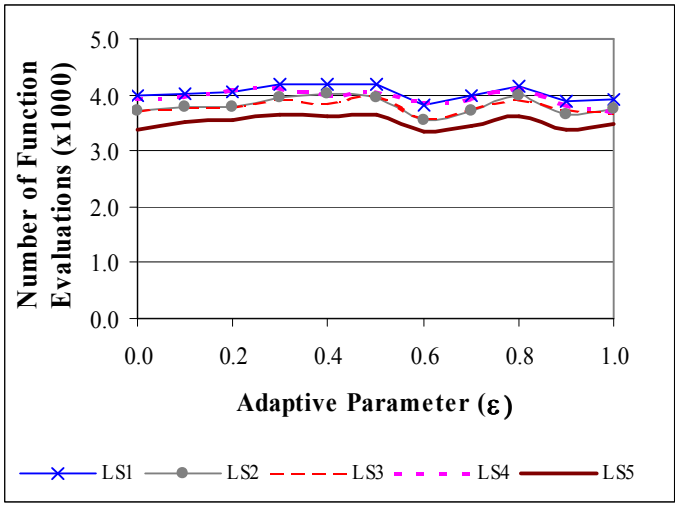



**Fig. 3.8 Effects of adaptive parameter**

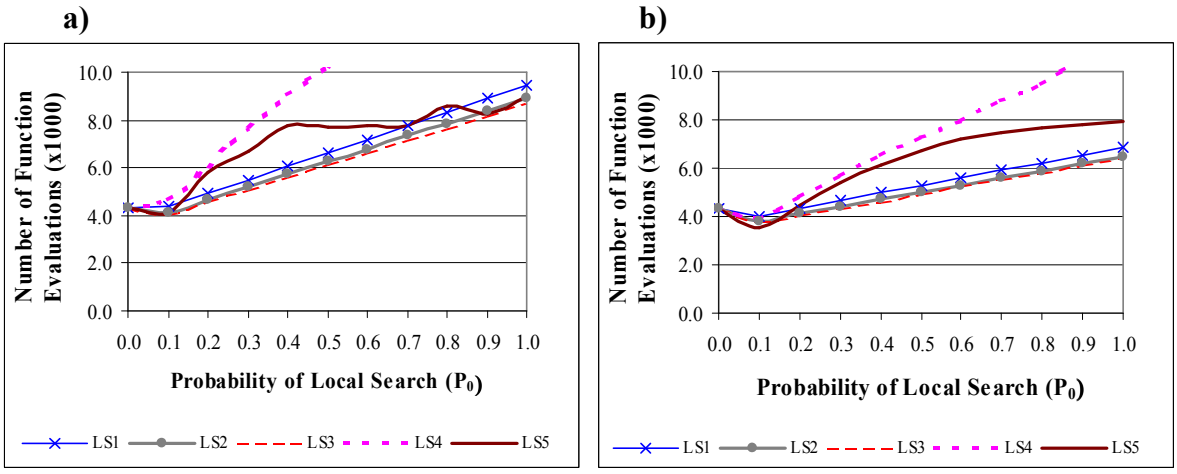



**Fig. 3.9 Probability of local search: a) NAHGA, b) SAHGA**

search is zero (this is equivalent to the SGA) for most of the local search algorithms in study. For LS3 (evolutionary strategy) and LS5 (steepest descent), the optimal performance is achieved when the probability of local search is close to 10%. On the other hand, SAHGA achieves optimal or very near optimal performance for a broader range of initial probabilities of local search (between 0.05 and 0.2) due to its adaptation of $P_0$ during the run. Another difference

between SAHGA and NAHGA is that, for increasing $P_0$, the number of function evaluations increases faster with NAHGA than with SAHGA, as shown in Fig. 3.9. The number of function evaluations increases much faster for LS4 and LS5 because they are the only local search algorithms that require more than one function evaluation per local search. This behavior was also observed with the other 7 test problems (see Appendix A, Figs. A.2, A.7, A.12, A.17, A.22, A.27 and A.32).

**c) Local Search Iterations.** The next experiment analyzes the number of iterations in the local search step. Fig. 3.10 shows the results of this experiment. These results indicate that the total number of function evaluations for the NAHGA algorithm increases with the number of local search iterations allowed (for the 5 local search algorithms), but for the SAHGA algorithm the number of function evaluations remains constant because of the adaptive stopping criterion in the SAHGA local search algorithm. These results show the advantage of the adaptive local search approach for improving performance. Similar results were also observed for the other test problems (see Appendix A, Figs. A.4. A.9, A.14, A.19, A.24, A.29 and A.34).

**d) Evolution: Baldwinian versus Lamarckian.** The final parameter to study is the proportion of Baldwinian evolution relative to the number of individuals undergoing local search. Fig. 3.11 shows the results for both algorithms. From the results, it is clear that SAHGA is almost insensitive to the value of this parameter, as long as some Baldwinian evolution is used, and NAHGA's performance is more sensitive. These results again show that the adaptive nature of SAHGA makes the algorithm more robust and reliable. Similar results were also observed for

the remaining test problems (see Appendix A, Figs. A.5, A.10, A.15, A.20, A.25, A.30 and A.35).

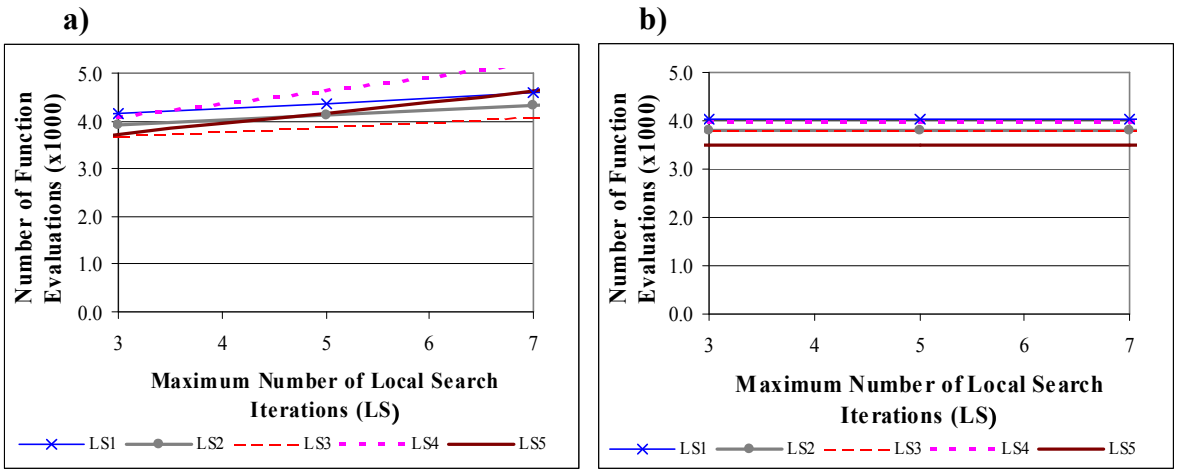



**Fig. 3.10 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

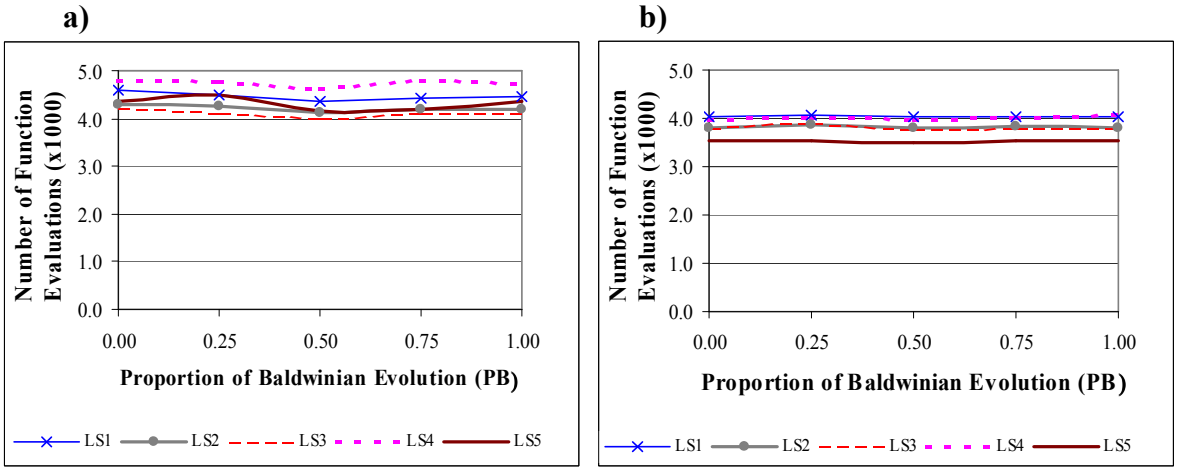



**Fig. 3.11 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

### *3.5.2 Algorithm Performance*

The overall performance of the algorithm is presented in detail for Griewank function, followed by a summary of performance on the 8 test functions. For detailed results for the other test functions, please see Appendix A.

**a) Detailed Performance for Griewank Function.** The evolution of best fitness, convergence ratio, average number of function evaluations, and reliability are discussed below. The first 2 results were evaluated using the default parameters for one initial population. The first graph (Fig. 3.12) plots the best fitness in the population during the SGA and SAHGA search processes, with SAHGA using the 5 local search techniques. Only the results attained by the application of SAHGA are included because SAHGA performance is consistently better than NAHGA for the 8 test problems.

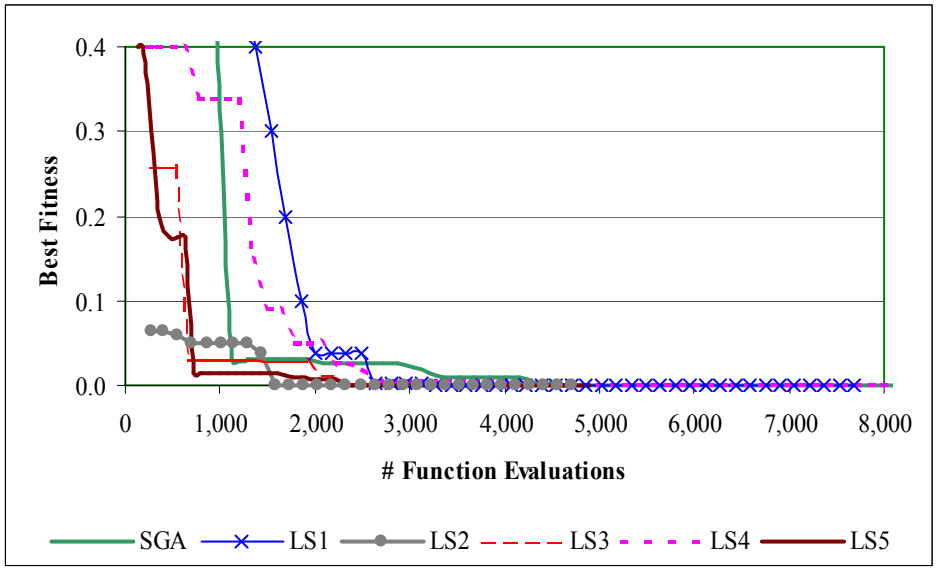



**Fig. 3.12 Best fitness for Griewank**

Fig. 3.12 shows that the SGA is inferior (in terms of performance) to SAHGA with all five local search algorithms. SGA required the most function evaluations with a total of 8,800. Steepest descent (LS5) performed the best, with only 4,896 function evaluations with a population size of 64, a reduction of 44% over the SGA (for one realization). The steepest descent method's speed and its ability to reduce the required population offset the fact that it is the most expensive method in terms of function evaluations per iteration.

The second result, shown in Fig. 3.13, is the convergence ratio, which is defined as the number of individuals with fitness equal to or near the fitness of the best individual. Fig. 3.13 shows that the results attained by SAHGA are again better than the SGA for all 5 local search approaches.

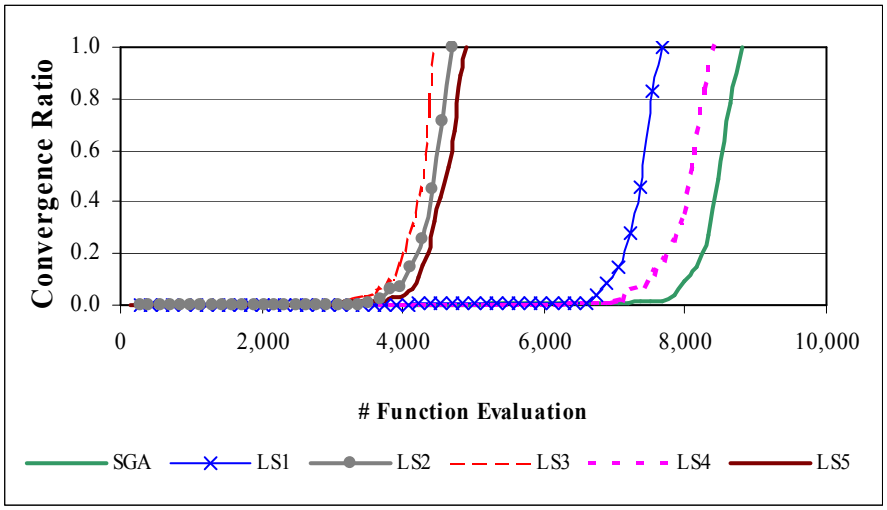



**Fig. 3.13 Convergence ratio for Griewank**

The solution quality of the algorithms was also evaluated by performing 100 runs with different random seeds, creating different initial populations, for every one of the parameter combinations given in Section 3.6.1. Table 3.6 and Fig. 3.14 and 3.15 show the results of these simulations, which tested the robustness of the algorithms under different starting conditions. The results are presented for the best, default and worst parameter combinations found in Section 3.5.1. For SAHGA, the average number of function evaluations for the default and best set of parameters are always better than the results attained with the SGA (ranging from 7% to 19% improvement), and the difference between the best set of results and the default varies between 2% and 8%. On the other hand, for NAHGA sometimes the results associated with the default set of parameters are worse than the results attained with the SGA, and the difference

between the best set of results and the default set of results varies between 10% and 25% for different local search algorithms. This shows that parameter estimation is critical to attaining good performance for the NAHGA, but the adaptive capabilities of SAHGA reduce the need for careful parameter selection. For both algorithms, the worst set of results was attained when all of the individuals are undergoing local search.

**Table 3.6 Monte Carlo Simulation Results: Average Number of Function Evaluations**

| | SGA | SAHGA | | | NAHGA | | |
|---|---|---|---|---|---|---|---|
| | | Worst | Default | Best | Worst | Default | Best |
| **LS1** | 4,326 | 6,851 | 4,020 | 3,906 | 9,462 | 4,373 | 3,936 |
| **LS2** | 4,326 | 6,470 | 3,800 | 3,720 | 8,936 | 4,131 | 3,717 |
| **LS3** | 4,326 | 6,402 | 3,775 | 3,660 | 8,723 | 4,036 | 3,478 |
| **LS4** | 4,326 | 13,776 | 3,964 | 3,666 | 14,337 | 4,645 | 3,508 |
| **LS5** | 4,326 | 7,905 | 3,503 | 3,292 | 8,961 | 4,153 | 3,071 |

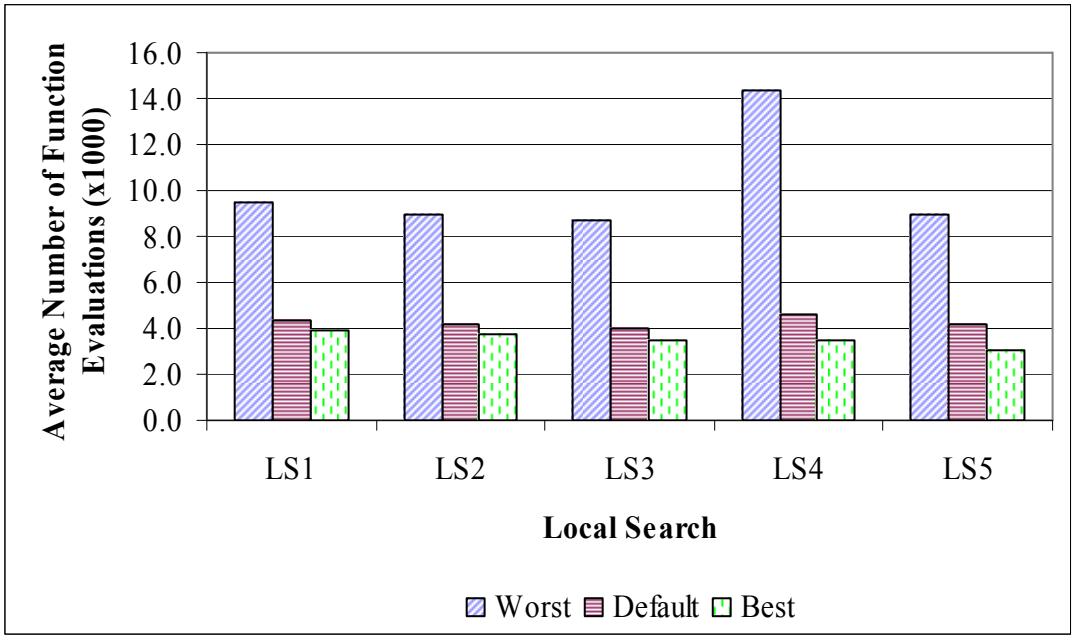



**Fig. 3.14 Monte Carlo simulation results: NAHGA**

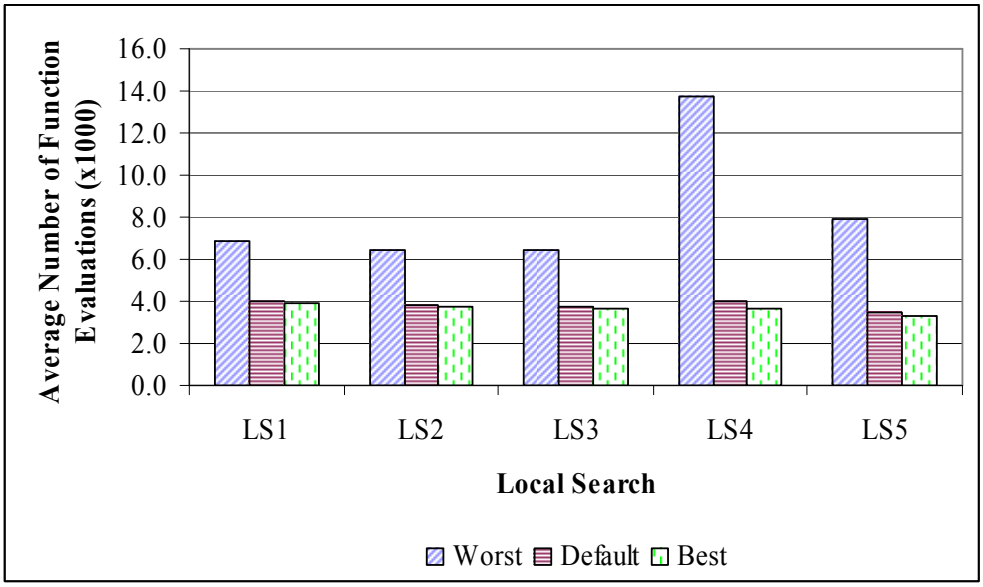



**Fig. 3.15 Monte Carlo simulation results: SAHGA**

Finally, Fig. 3.16 shows the reliability of SGA, SAHGA and NAHGA for the default set of parameters, where reliability is measured in terms of the percentage of the 100 runs that identified the optimal solution. SAGHA reliability was in the range of 90-100%, while solving the problem over a much smaller population than the SGA, which means it is less computationally expensive. NAHGA is very efficient as well, with reliability values of 90% to 100%, again with smaller population sizes than the SGA. SAHGA and NAHGA have somewhat lower reliabilities for LS4 and LS5 because the extensive standard deviation reduction associated with the gradient-based algorithms sometimes causes premature convergence due to loss of diversity in the population.

**b) Performance Summary for All Test Functions.** Tables 3.7 and 3.8 present a summary of the results for all eight test functions in terms of average savings and reliability for 100 realizations, respectively.

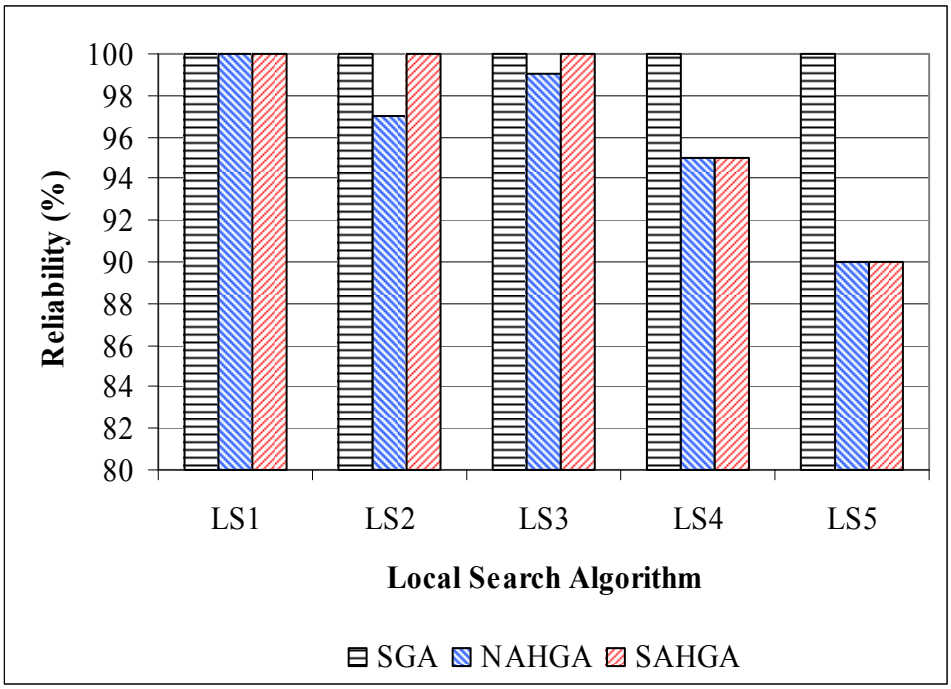



**Fig. 3.16 Reliability**

**Table 3.7 Average Savings for SAHGA (%)**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 24.5 | 24.5 | 24.5 | 12.1 | -12.6 |
| **DJ2** | 17.6 | 17.6 | 17.6 | 29.5 | 25.1 |
| **Branin** | 31.8 | 38.0 | 38.8 | 46.7 | 45.2 |
| **Six-Hump** | 29.8 | 29.8 | 29.8 | 29.3 | 29.1 |
| **Schwefel** | 8.9 | 10.4 | 12.3 | 31.5 | 21.2 |
| **Griewank** | 7.1 | 12.2 | 12.7 | 8.4 | 19.0 |
| **Test08** | 24.2 | 27.2 | 28.3 | 22.7 | 27.4 |
| **B&M** | 8.3 | 13.4 | 13.0 | 34.2 | 33.2 |

Table 3.7 shows that the SAHGA algorithm performs better than SGA for all the test functions and local search algorithms except LS5 with function DJ1. For the eight test problems, the savings are from 7% to 47%. In general, the savings are higher for random derivative and gradient, even when these local search algorithms are more expensive. This phenomenon is related to the differences in population size of each algorithm for different test functions. For

example, when the differences in population size between steepest descent and random search is not significant (e.g., see functions DJ1 and Six-Hump in Table 3.5), random search is more likely to perform as well as or better than steepest descent given the additional computational effort necessary for gradient-based algorithms. On the other hand, when the population size varies substantially across different local search methods (see Griewank, Schwefel, or B&M functions in Table 3.5), steepest descent is more likely to perform better than random walk.

**Table 3.8 Reliability for SAHGA (%)**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 100 | 100 | 100 | 98 | 97 |
| **DJ2** | 100 | 100 | 100 | 95 | 90 |
| **Branin** | 100 | 100 | 100 | 100 | 98 |
| **Six-Hump** | 100 | 100 | 100 | 95 | 95 |
| **Schwefel** | 100 | 100 | 100 | 95 | 90 |
| **Griewank** | 100 | 100 | 100 | 95 | 90 |
| **Test08** | 100 | 100 | 100 | 98 | 95 |
| **B&M** | 100 | 100 | 100 | 95 | 90 |

## 3.6 Conclusions and Recommendations

The results presented indicate that the local search capabilities of the HGA algorithm enabled robust solution of complex, unimodal, multi-modal, and constrained problems with less effort than the SGA. Of the two hybrid algorithms, SAHGA is always superior to NAHGA because near-optimal performance is attained for a broad range of parameters. For both algorithms, local search is best when used sparingly. Too much local search increases computational effort with diminishing returns in search efficiency.

Another result is related to the importance of the local search algorithm selected. The results show that the best performance is not always attained for the same local search algorithm across all test functions, mostly because of the difference in effort necessary to apply the different local search algorithms. For any function, the most suitable local search algorithm can be pre-selected using only the population sizing analysis shown in Table 3.5. When the difference in population sizes for different local search algorithms is not significant, it is more likely that a random search algorithm (which requires only 1 function evaluation per local search iteration) will perform better than steepest descent (which requires $n$+1 function evaluations per iteration, where $n$ is the number of decision variables) or random derivative (which requires 2 function evaluations per iteration).

Another important result derived from this research is related to the population sizing approach presented in Section 3.4. This process allows us to evaluate a good estimate of the population to achieve optimal performance, but there is a possibility that the same level of reliability could be achieved with a smaller population. With respect to the HGA, the analysis presented shows clearly how local search reduces the standard deviation of fitness in the population, which in turn reduces the required population size to achieve the same level of reliability. This reduced population together with the HGA operations reduces the total number of function evaluations by 5% to 44% on average, for every one of the problems in study.

# Chapter 4

## APPLICATION TO GROUNDWATER REMEDIATION DESIGN TEST PROBLEM

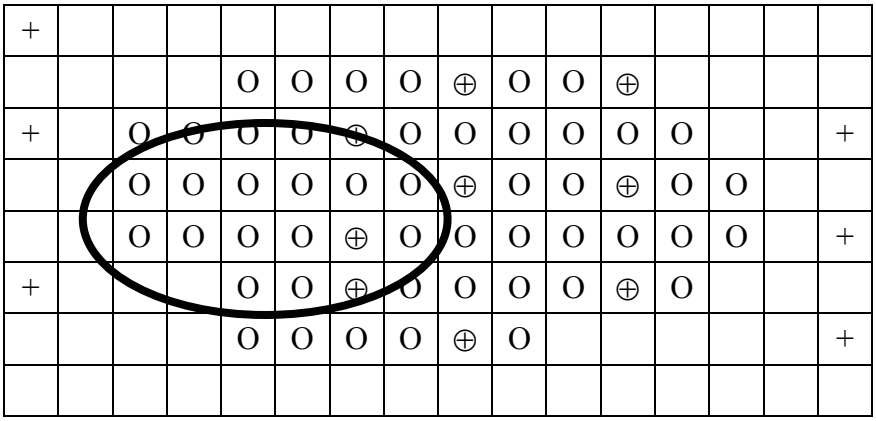
The results in the previous chapter show substantial promise for improved performance using hybrid GAs, but only for test functions. This chapter presents a groundwater remediation design application that is then used to test performance of NAHGA and SAHGA on a real–world problem. Section 4.1 presents the test problem. Section 4.2 presents the characteristics of the local search algorithm selected for this application, the population sizing, and a solution approach designed to minimize the number of function evaluations in the solution of this problem. Section 4.3 presents the results. Finally, Section 4.4 presents the conclusions and recommendations for future applications of this algorithm to real world problems.

### 4.1 Groundwater Remediation Design Test Problem

#### *4.1.1 Description*

The performance of the algorithms was assessed in the design of a risk-based groundwater remediation system for a hypothetical aquifer similar to the case studied by Smalley et al. (2000) and Gopalakrishnan et al. (2003). The aquifer is heterogeneous, isotropic and confined. It is assumed to be contaminated with BTEX (benzene, toluene, ethylbenzene, and xylene) with a peak initial concentration of 200 mg/L. The problem considers minimization of the pumping cost in order to meet a specific human health risk target. The treatment technology is assumed to be pump-and-treat. The pumping system is designed with up to three wells with a maximum

capacity of 250 m³/day each. The location of these 3 wells can be selected among 58 predetermined locations. Among these 58 wells, 15 are preexisting and are used as observation wells. The treatment technology for the extracted water is assumed to be air stripping. Fig. 4.1 presents the aquifer system with the location of the observation and potential extraction or injection wells.



+ Observation wells (15)
O Possible Remediation wells (58): injection or extraction

**Fig. 4.1 Plan view of the case study aquifer**

The dimensions of the area of study are 240 x 480 m. For computational speed to perform the test runs, the area was modeled using finite-difference method with a coarse grid of 8 by 16 elements (each 30 x 30 m). The hydraulic conductivity distribution was derived from a distribution for a finer mesh of 128 by 64 elements. The hydraulic conductivity generation technique is detailed by Smalley et al. (2000). The flow and contaminant transport were modeled using the computer models MODFLOW (McDonald and Harbaugh, 1988) and RT3D (Clement 1997; Clement et al., 1998, 2000). The transport step includes advection, dispersion and linear adsorption. The risk was evaluated from the predicted concentrations at the observation wells

shown in Fig. 4.1 using the analytical model presented by Smalley et al. (2000). The characteristics of the aquifer are presented in Table 4.1.

**Table 4.1 Aquifer Characteristics**

| Parameter | Value |
|---|---|
| Geometric mean of hydraulic conductivity K (m/day) | 6.18 |
| Geometric standard deviation of hydraulic conductivity K (m/day) | 0.54 |
| Porosity n | 0.3 |
| Longitudinal dispersivity $\alpha_L$ (m) | 15.00 |
| Transversal dispersivity $\alpha_T$ (m) | 3.00 |
| Aquifer thickness b (m) | 20.00 |
| Retardation coefficient R | 1.41 |

For this problem, the designs are represented by 4 decision variables: pumping well location, pumping rate ($u_i$), injection or extraction ($Y_i$), and installation ($X_i$). Variables $X_i$ and $Y_i$ are given by:

$$X_i = \begin{cases} 1 & \text{if well is installed at location i} \\ 0 & \text{Otherwise} \end{cases}$$
$$Y_i = \begin{cases} 1 & \text{for injection from well i} \\ 0 & \text{for extraction from well i} \end{cases} \qquad (4.1)$$

The cost of each design is evaluated by the following objective function (Smalley et al., 2000):

$$\text{Min } C_{TOT} = C_{REM}(u_i) + C_{MON}(X_i) + C_{SYST}(u_i) \qquad (4.2)$$

where the total cost $C_{TOT}$ consists of three components – $C_{REM,}$ which is the capital and operating costs for the wells; $C_{MON,}$ which is the cost of on-site monitoring; and $C_{SYST,}$ which includes additional capital and operating costs for the ex-situ treatment system. Extensive details on the remediation and monitoring cost are presented by Smalley et al. (2000). The final component

($C_{SYST}$) was evaluated from data obtained from RACER (1999), a parametric cost modeling system, as detailed by Gopalakrishnan et al. (2003).

The optimal design must satisfy human health risk limits, pumping rate limits, and limits on hydraulic drawdowns. The constraints are presented in Eqs. (4.3) to (4.5). The risk constraint follows,

$$\mathrm{Risk}_{t,k}^{\mathrm{TOTAL}} = \mathrm{Risk}_{t,k}^{\mathrm{w}} + \mathrm{Risk}_{t,k}^{\mathrm{shw}} + \mathrm{Risk}_{t,k}^{\mathrm{nc}} \leq \quad \mathrm{TR}\ \forall t, \forall k \tag{4.3}$$

where $\mathrm{Risk}_{t,k}^{\mathrm{w}}, \mathrm{Risk}_{t,k}^{\mathrm{shw}}$ and $\mathrm{Risk}_{t,k}^{\mathrm{nc}}$ are the cancer risks due to ingestion of contaminated drinking water, inhalation of volatiles from contaminated water due to showering, and inhalation of volatiles from contaminated water due to other non-consumptive uses, respectively (see Smalley et al, 2000, for details on how these terms are calculated). The human health risk constraint ensures that the total lifetime carcinogenic risk, $\mathrm{Risk}_{t,k}^{\mathrm{TOTAL}}$, which is evaluated for different times *t* and exposure locations *k*, is less than the target risk level, TR.

Eqs. (4.4) and (4.5) represent limits on pumping rates and hydraulic heads, where $u_{min,i}$ and $u_{max,i}$ represent the minimum and maximum pumping rates for a given remediation well *i* ($m^3$/day); $h_{i,l}$, $h_{min,l}$, and $h_{max,l}$ are the computed hydraulic head for remediation well *i* (m), the minimum hydraulic head (m), and the maximum hydraulic head (m) allowed at remediation well location *l*, respectively.

$$\mathrm{u}_{\mathrm{min,i}} \leq \left| \mathrm{u}_{\mathrm{i}} \right| \leq \mathrm{u}_{\mathrm{max,i}} \qquad \forall \mathrm{i} \tag{4.4}$$

$$\mathrm{h}_{\mathrm{min,l}} \leq \mathrm{h}_{\mathrm{i,l}}(\mathrm{u}_{\mathrm{i}}) \leq \mathrm{h}_{\mathrm{max,l}} \qquad \forall \mathrm{i}, \forall \mathrm{l} \tag{4.5}$$

Finally, the fitness of the design is evaluated by combining the cost (from Eq. (4.2)) with penalties for violations of the risk and head constraints in Eqs. (4.3) and (4.5). No penalties are necessary for pumping rate limits in Eq. (4.4) because the pumping rate is one of the decision variables and it is limited directly in the GA through its representation as a binary number. The final fitness equation (with linear penalty) is presented in Eq. (4.6):

$$\text{Fitness} = C_{TOT} + \omega_1 \cdot \text{Risk} \quad \text{violation} + \omega_2 \cdot \text{Head} \quad \text{violation} \tag{4.6}$$

where $\omega_1$ and $\omega_2$ are the penalty weights for the risk and head constraints respectively. For this case study, the values of $\omega_1$ and $\omega_2$ have been set to 1000 by means of a trial-and-error process. In this process, the penalty weights are incrementally increased, by a factor of 10, from a base value of 1 until a feasible solution to the problem is found. Note that for this application, fitness is minimized so low fitness designs are optimal.

### *4.1.2 Complexity Analysis*

The complexity of this problem can be shown by evaluating the total number of designs that can be created with 58 pumping well locations. The analysis assumes that the pumping rates can be chosen in increments of 1 $m^3$/day starting at 1 and ending at 250 $m^3$/day (the same increments used by the SGA). At the same time, because the wells can inject or extract water from the aquifer, the total number of possible pumping rates is equal to 500 per well. Using this information, the total number of possible designs for this simple problem is equal to 3.9 trillion, as calculated in Table 4.2. For this reason, GAs are needed to explore the decision space of this problem in a more effective way than trial-and-error search.

**Table 4.2 Total Number of Combinations**

| # Wells | Well Combinations | Pumping Rates Combinations | Total Combinations |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 58 | 500 | 58 x 500 |
| 2 | 1,653 | $500^2$ | 1,653 x $500^2$ |
| 3 | 30,856 | $500^3$ | 30,856 x $500^3$ |
| **Total** | | | **$\approx 3.9 \cdot 10^{12}$** |

## 4.2 Hybrid Genetic Algorithm (HGA)

### *4.2.1 Local Search Algorithm*

For this application, the local search algorithm used is random walk with uniform distribution (LS1) presented previously in Section 3.4. This algorithm was selected because it performed consistently on the test functions in the previous chapter and required low computational effort.

### *4.2.2 Population Size*

The population size for the GA analysis was evaluated using the methodology presented in section 3.5. The standard deviation reduction attained with this algorithm for this problem is presented in Fig. 4.2. From this plot, the parameter $\beta$ from Eq. (3.8) was chosen to be 0.83, as the average over the first 5 local search iterations. Following the analysis previously presented, the parameters $\sigma_f$ and d were evaluated to be 1,404,403 and 157,849, respectively. Using this information, the initial population for the SGA was estimated from Eq. (3.7) to be 30 individuals for K=1. For the HGA, the initial population size was estimated from Eq. (3.8) to be 20 for K=1. Table 4.3 shows the different population sizes for different values of parameter K after the application of Eqs. (3.7) and (3.8).

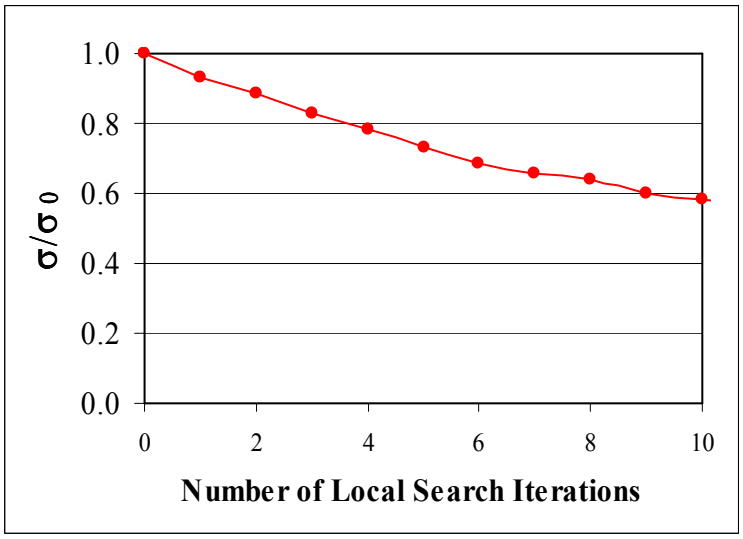



**Fig. 4.2 Standard deviation reduction (β)**

**Table 4.3 Population Size for SGA and HGA**

| Population Order K | Population Size | |
|---|---|---|
| | SGA | HGA |
| 1 | 30 | 25 |
| 2 | 60 | 50 |
| 3 | 120 | 100 |
| 4 | 240 | 200 |
| 5 | 480 | 400 |

### *4.2.3 Injection Approach*

In order to reduce the computing time, an elitist serial solution process was developed. In this new approach, the problem is first solved for the base population (K=1). The next step is usually to solve the problem for the next population (K=2) starting from a random population (Reed et al., 2000). In the new approach, this step is modified by randomly replacing one of the individuals in the new population with the best individual from the previous population. Using

this approach with $\mu+\lambda$ selection, the search is guaranteed to always improve the solution for each subsequent value of K, or in the worst case scenario the solution does not change. This process is an adaptation of the multiobjective approach presented by Reed (2002) and the concepts of continuation theory presented by Srivastava and Goldberg (2001) and Srivastava (2002). This new approach is called the "injection" method and the traditional approach is called the "no-injection" method.

This "injection" approach is somewhat similar to the micro genetic algorithm ($\mu$GA) proposed by Krishnakumar (1989). In the $\mu$GA approach, a small population of individuals is evolved until the entire population converges to the same solution. Then all the individuals but one are re-initialized and the process starts again until a new solution is attained. The difference between these two approaches is in the population sizes used in successive runs. For the $\mu$GA, the population is constant and arbitrarily defined. On the other hand, for the injection approach, the population is doubled every time the algorithm restarts and the initial population is chosen based on population sizing theory.

A comparison of the injection/no-injection performance is presented in Fig. 4.3 for SAHGA. In the line for injection, the symbol represents the injection point. From this plot, it is clear that the injection method solves the problem faster because it is using the information generated by the previous population to speed up the search. On the other hand, the no-injection method starts the search every time from a random population. In fact, the injection technique needs approximately 9,500 function evaluations to find the solution and the no-injection method

needs approximately 17,400. This is equivalent to 45% savings in the number of function evaluations required for convergence.

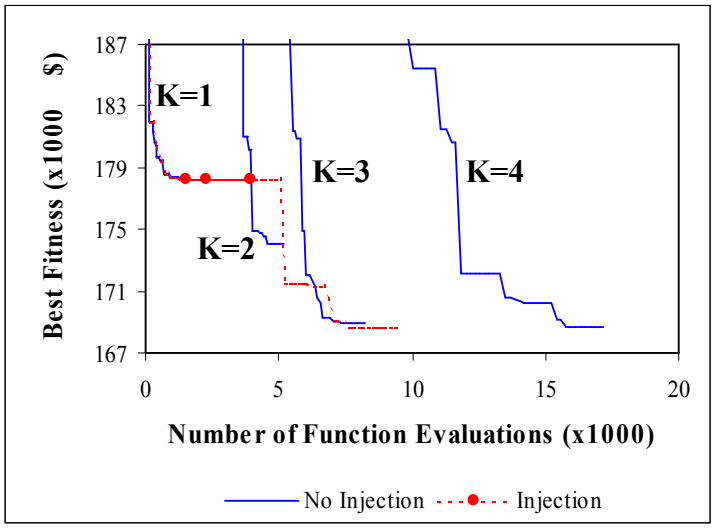



**Fig. 4.3 Comparison of Injection/No-Injection methods for SAHGA**

## 4.3 Experimental Results

Two types of experiments were performed to evaluate the HGAs (SAHGA and NAHGA): parameters and performance. The first experiments were performed to compare the results for the different parameters involved in the HGA analysis: local search frequency, probability of local search, number of local search iterations, adaptive parameter, and proportion of Baldwinian evolution. The second category of results compares the HGA algorithms' overall performance to the SGA.

### *4.3.1 Parameter Analysis*

For the analysis of the performance of the algorithms with respect to the different parameters, a default set of parameters was selected based on previous experience working with

test functions (See Chapter 3). To save computational effort, the parameter analysis was performed for a fixed population of 40 individuals (K=2 in Table 4.3), which gave satisfactory results with the default parameters. The performance of the algorithms was measured using the time, in terms of the total number of function evaluations, to converge to the best solution found.

**a) Local Search Frequency.** The first experiment was designed to evaluate the effect of local search frequency on the solution of the problem. For the NAHGA algorithm, local search was performed at a pre-defined interval $\Delta G$; for the SAHGA algorithm, local search followed the threshold requirements previously explained. Fig. 4.4 a) and b) show the results for the NAHGA and SAHGA algorithms, respectively. For NAHGA, it is clear that the optimal results are achieved only for one value of the variable in study, $\Delta G=4$ (meaning that local search is performed every four generations). On the other hand, for the SAHGA the optimal results are achieved for a set of different values of T, so the algorithm is more robust. These SAHGA results are consistent with the results presented in Chapter 3 for the test problem Griewank (see Fig. 3.7) and in Appendix A for the other 7 test problems. However NAHGA is much more sensitive to the local search frequency for this application than for the test function in Chapter 3 (compare Fig. 3.7 a) with Fig. 4.4 a). One plausible explanation for this difference is that the test functions from Chapter 3 were either unconstrained or unimodal and constrained, and this problem is multimodal and constrained. At the beginning of the optimization run, there are few or no feasible individuals and the local search capabilities of SAHGA and NAHGA help increase the number of feasible individuals in the population. In the case of NAHGA, the frequency of local search then becomes critical to the performance of this process. The adaptive capabilities of SAHGA make it more robust to these types of effects.

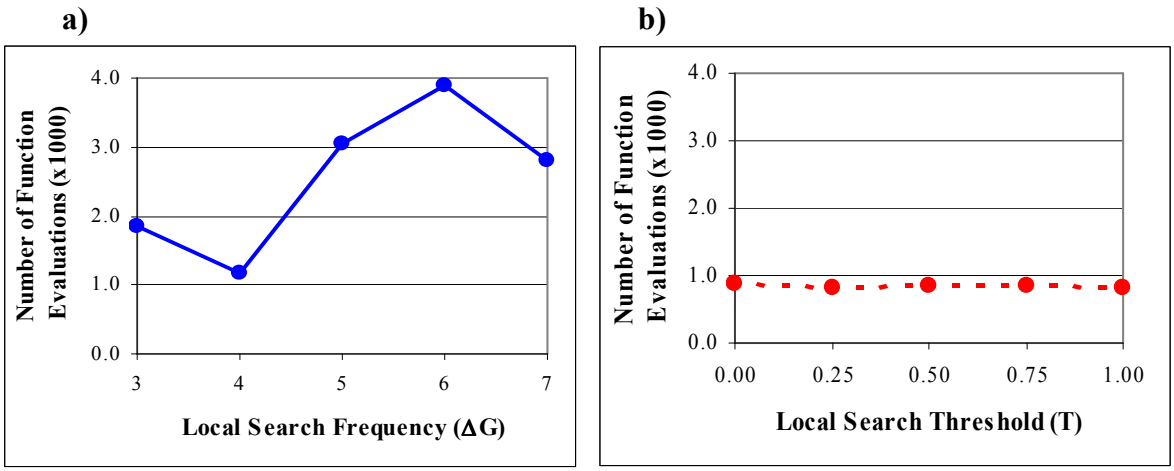



**Fig. 4.4 Local search effect for NAHGA algorithm (a) and SAHGA algorithm (b)**

**b) Local Search Probability and Adaptive Parameter.** The second experiment tested the effect of the probability of local search parameters on performance. In SAHGA, the probability of local search is adapted using the parameter ε in Eq. (3.2). Fig. 4.5 a) shows the effect of the adaptive parameter for the default initial probability of local search. This experiment shows that there is some improvement in performance for different values of the adaptive parameter. The best performance is attained for ε=0.8, with 390 function evaluations, but the performance when ε=1 is nearly the same (428 function evaluations). Performance in the range of ε=0.3 to 0.7 is also quite good. A value of ε=1 means that only 1 local search iteration is performed (see Eq. (3.2)). This result shows that the first iteration of local search is the most essential for this application and adaptation may not always be necessary. These results are similar to the behavior observed for the test problems from Chapter 3, but with somewhat more sensitivity to low values of ε (compare Fig. 4.5 a with Fig. 3.8).

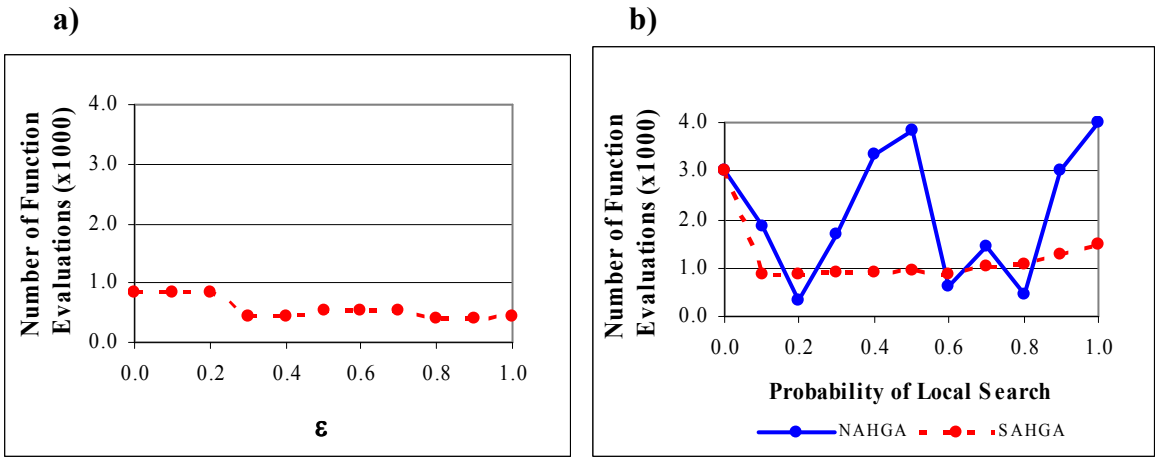



**Fig. 4.5 Effects of adaptive parameter (a) and probability of local search (b)**

The next parameter to study is the probability of local search, which determines the number of individuals undergoing local search. Fig. 4.5 b) shows the results for NAHGA and SAHGA (for the default value of ε). From these results, it is clear that the optimal performance of NAHGA for these default parameter settings is when the probability of local search is 0.2. On the other hand, SAHGA achieves optimal or very near optimal performance for a broader range of initial probabilities of local search (between 0.1 and 0.8) due to its adaptation of $P_0$ during the run. These results are similar to the test functions for SAHGA for all values except $P_0 = 0$ (no local search), which shows much worse relative performance than the test functions. NAHGA shows much more sensitivity for this application, however, with vastly different performance for different values of $P_0$ (compare Fig. 4.5 b for LS1 and Fig. 3.10). As with the local search frequency, these differences are likely due to the multimodality and constrained nature of this application problem versus the unconstrained test problems.

**c) Local Search Iterations.** The next experiment analyzes the number of iterations in the local search step. Fig. 4.6 shows the results of this experiment. These results indicate that the

total number of function evaluations for the NAHGA algorithm increases with the number of local search iterations allowed, but for the SAHGA algorithm the number of function evaluations remains constant because of the adaptive stopping criterion in the SAHGA local search algorithm. These results show the advantage of the adaptive local search approach for improving performance, similar to the test problems presented in Chapter 3 (see Fig. 3.10).

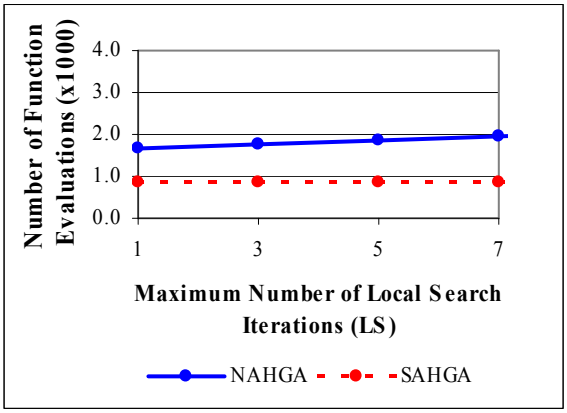



**Fig. 4.6 Effect of maximum number of local search iterations on SAHGA and NAHGA performance**

**d) Evolution: Baldwinian v Lamarckian.** The final parameter to study is the proportion of Baldwinian evolution relative to the number of individuals undergoing local search. Fig. 4.7 shows the results for both algorithms. The results show that SAHGA's performance is best when only Baldwinian or Lamarckian evolution is used, and the worst performance is attained when these two evolution approaches are equal. NAHGA's performance shows much greater variation for different values of the parameter. This result again shows that the adaptive nature of SAHGA makes the algorithm more robust and reliable. These results show somewhat more sensitivity to the value of PB than for the test functions (see Fig. 3.11), especially for NAHGA.

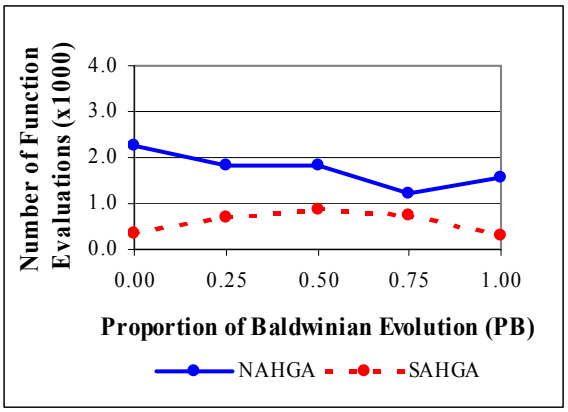



**Fig. 4.7 Effect of proportion of Baldwinian evolution on SAHGA and NAHGA performance**

### *4.3.2 Overall Algorithm Performance*

This section compares the overall performance of the HGA relative to the SGA using the multipopulation injection solution approach described in the methodology section. In this section, SAHGA was evaluated for a single random initial population using the default set of parameters, which showed good performance in the previous section. However, from the previous section, it is clear that the selection of the parameters for NAHGA is important for optimal performance, so 10 trial-and-error iterations were performed to identify a good set of parameters. In every one of these steps, the NAHGA algorithm was evaluated for at least 20,000 function evaluations before a new set of parameters was selected. The final set of parameters for the NAHGA analyses in this section were: $\Delta G=3$, LS=1, P=0.2 and PB=0.5. For this set of parameters, convergence was attained in approximately 6,000 function evaluations, but the solution found was approximately 2% worse than the solution attained with SAHGA. No other set of parameters was found for NAHGA that could attain the best solution found with SAHGA.

Fig. 4.8 shows the best fitness in the population during the runs for each algorithm. In this figure, x's represent the moment when the best solution is found for the first time. SAHGA finds the solution for the first time after 8,000 function evaluations while NAHGA needs 5,000 (but, as noted above, does not find the optimal solution) and the SGA needs 33,000 evaluations. SAHGA identified the best solution more slowly than the SGA, due to the extra function evaluations required for local search, but its local search capabilities enabled it to converge to the correct solution faster once it was identified. In fact, SAHGA took approximately 9,500 function evaluations to completely solve the problem, when the best solution took over the population for two consecutive population sizes, versus 40,000 for SGA, which is a 75% improvement. On the other hand, NAHGA took a shorter time, but the solution quality is not as good as the one attained by SAHGA, and the necessary trial-and-error process for parameter evaluation makes the application of NAHGA impractical.

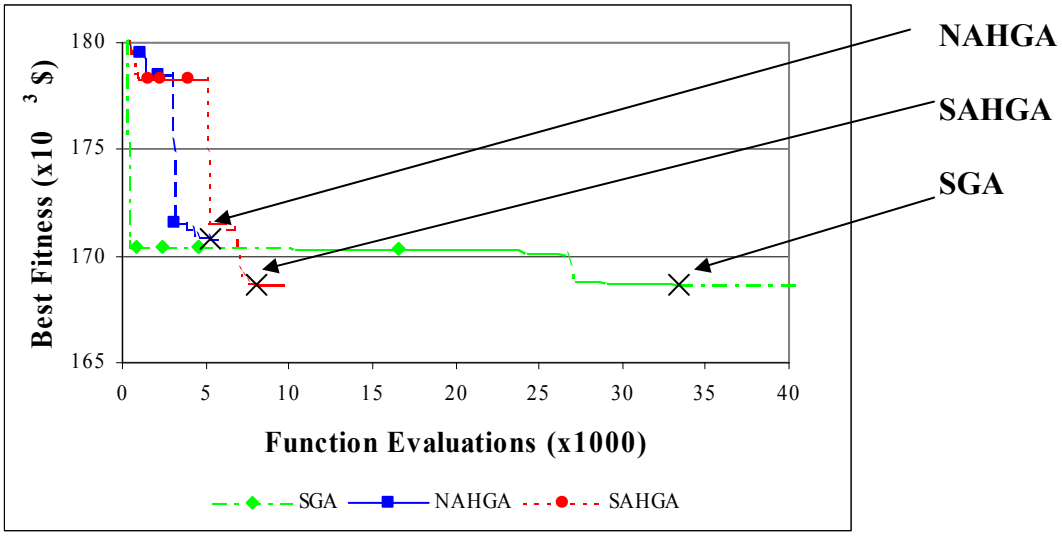



**Fig. 4.8 Performance for SGA, NAHGA and SAHGA**

More important than the speed of identification of the first individual discussed above, which can be influenced by random chance, is the rate of convergence to the optimal solution. Fig. 4.9 shows the convergence ratio (CR) for each algorithm, which is defined as the number of individuals with fitness equal to or close enough to (when the difference between the best individual and the sub-optimal individual is less than $10^{-5}$) the best fitness in each generation. From this plot, it is clear that the best solution takes over the population substantially faster in SAHGA and NAHGA than in the SGA because local search is providing more information that speeds up the search. SAHGA and NAHGA achieve complete convergence in 75% and 85% fewer function evaluations than the SGA, respectively. Recall that this result was achieved for only one initial population size. Results for a range of initial populations are explored in the next chapter.

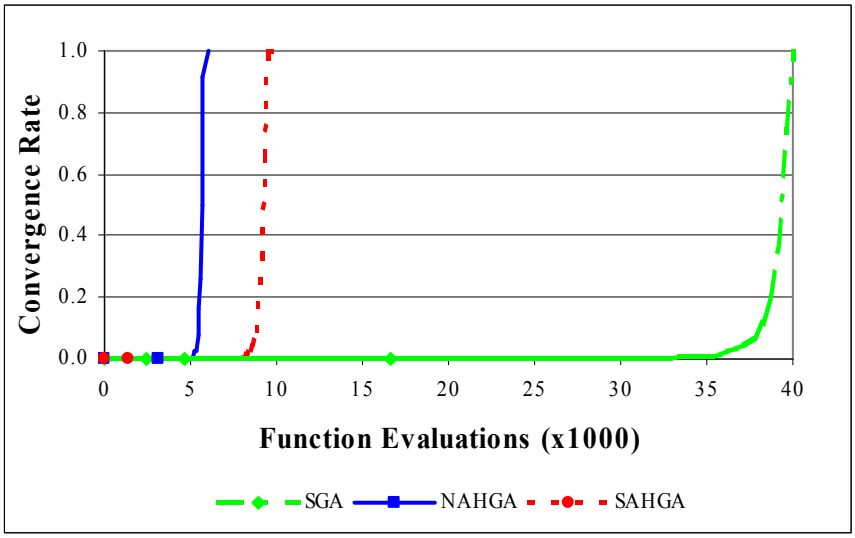



**Fig. 4.9 Convergence ratio for SGA, NAHGA and SAHGA**

Finally, Table 4.4 shows a comparison of the total number of function evaluations necessary for SAHGA and SGA to converge at each population size. For SAHGA, the

distribution between global and local search across the population is 90.6% (8,720/9627). That is, 90.6% of the function evaluations were spent doing global search. Of the 907 function evaluations spent on local search, only 150, or 16.5%, actually improve the solution because the random walk local search method is a heuristic approach that is not guaranteed to improve the solution. Despite this inefficiency in local search, SAHGA still achieves convergence 75% faster than SGA.

**Table 4.4 Number of Function Evaluations for SGA and SAHGA**

| | SGA | | | SAHGA | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **K** | **Pop** | **Final Generation** | **Total** | **Pop** | **Final Generation** | **Total** | **Global Search** | **Local Search** | |
| | | | | | | | | **Total** | **Effective** |
| 1 | 30 | 25 | 780 | 20 | 67 | 1474 | 1360 | 114 | 0 |
| 2 | 60 | 25 | 1,560 | 40 | 19 | 860 | 800 | 60 | 10 |
| 3 | 120 | 16 | 2,040 | 80 | 17 | 1552 | 1440 | 112 | 15 |
| 4 | 240 | 48 | 11,760 | 160 | 31 | 5741 | 5120 | 627 | 125 |
| 5 | 480 | 49 | 24,000 | | | | | | |
| **Total** | | | **40,140** | | | **9,627** | **8,720** | **907** | **150** |

## 4.4 Conclusions and Recommendations

The results presented in this paper clearly indicate that the local search capabilities of the SAHGA algorithm enabled better performance than the SGA, resulting in approximately a 75% reduction in the total number of function evaluations required for solution of a remediation design test case for a given initial trial population. These results are particularly impressive given that only 16.5% of the function evaluations spent on local search were effective in improving the solution. In comparison with the NAHGA algorithm, SAHGA is more robust because local

search is applied only when it is necessary and the performance does not change for a broad range of different parameter values.

It is important to note the differences and similarities between the results presented in this chapter and Chapter 3. For example, the parameter "T" (switch between local and global search for SAHGA) behaves in the same way for the groundwater remediation problem and the test problems from Chapter 3. Results are also similar for the number of local search iterations (LS) (for SAHGA and NAHGA), the adaptive parameter "ε" (for SAHGA), proportion of Baldwinian evolution (PB) and initial probability of local search "$P_0$". However, for the parameter "ΔG", switch between local and global search for NAHGA, performance is quite different. The behavior of NAHGA for the groundwater remediation problem is much more sensitive to the parameter value selected than the behavior for the other test problems. This difference can be explained by the constrained and multimodal nature of the groundwater remediation problem, which makes the local search frequency more critical for identifying feasible and globally optimal solutions.

# Chapter 5

## ANALYSIS OF ALGORITHMIC ENHANCEMENTS AND LOCAL SEARCH EFFECT ON SAHGA PERFORMANCE

Chapter 4 presented the application of the algorithm SAHGA to a groundwater remediation problem. In that chapter, the performance of the algorithm was evaluated by solving the problem for one initial population. The results showed that the behavior of the algorithm for different parameter configurations was usually similar to the behavior for the test problems analyzed in Chapter 3. After the evaluation presented in Chapter 4, a few questions remain:

(1) Are the savings achieved by the algorithm the same for different initial populations? The probable answer is negative because of the stochastic nature of the algorithm.

(2) Can we modify the algorithm to improve reliability and efficiency?

(3) What are the effects of different local search algorithms on performance?

In order to answer these questions, first the reliability of the algorithm is tested for different initial populations in Section 5.1. Second, a number of algorithmic improvements are evaluated in Section 5.2 for improving efficiency of the algorithm, especially in the selection of the individuals undergoing local search. Then, the effect of other local search algorithms on performance is evaluated in Section 5.3. The enhanced algorithm is then validated in Section 5.4 on a different set of initial populations to evaluate how general and effective are the proposed enhancements. Finally, Section 5.5 gives conclusions and recommendations.

## 5.1 Evaluation of Algorithm Reliability and Computational Effort

The objective of this section is to evaluate in more detail the performance and savings associated with SAHGA by solving the problem for 10 different initial populations independently. To ensure that the results are comparable, both algorithms (SGA and SAHGA) are run until the optimal solution is identified. If the previous condition is not satisfied for the first population (K=1), the problem is solved for the next population size (K=2) using the injection approach presented in Section 4.3.2. The process continues if necessary. Fig. 5.1 shows the results of this evaluation in terms of the total number of function evaluations. From the plot, it is clear that SAHGA is faster than SGA 9 out of 10 times (reliability 90%). The average savings are 18%. In the one case where SGA is faster than SAHGA (run 6), the negative savings occur because the SGA, through random chance, contained substantial portions of the final solution in the initial population. Because SGA and SAHGA have different initial population sizes at each building block order (K = 1, 2, etc.), the initial SAHGA population may not always contain the same portions of the final solution that the initial SGA population contains.

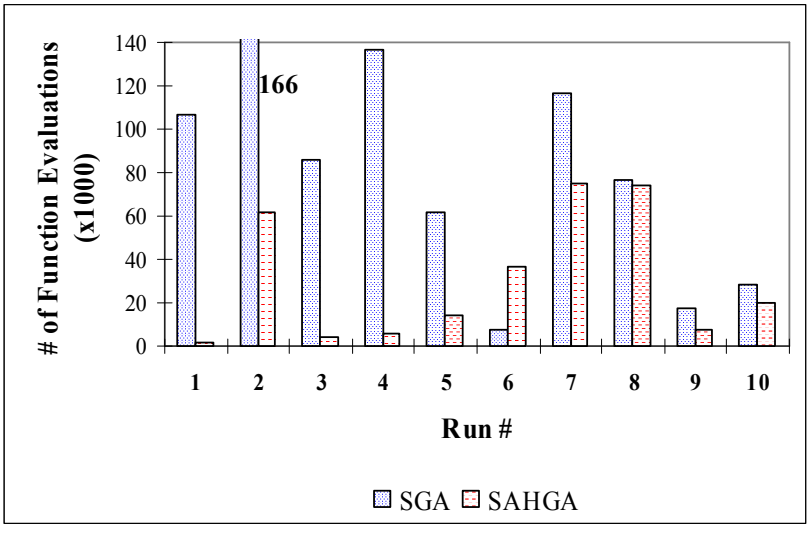



**Fig. 5.1 Comparison between SGA and SAHGA**

The results presented show that SAHGA requires 1,500 to 80,000 function evaluations to solve the problem, a rather large range in performance. This occurs because the algorithm converges quickly, but sometimes to sub-optimal solutions. For this reason, the injection approach sometimes needs to be applied up to K=5 (run 6) and sometimes only to K=1 or K=2 (runs 1 to 5 and 7). One reason for the large variance is that the groundwater remediation problem has numerous solutions that are nearly identical in objective function value to the optimal solution. From an engineering perspective, these solutions are sufficiently close to the optimal solution to be as good in practice as the optimal solution. Therefore, to avoid excessive computational requirements in testing numerous variations of the SAHGA algorithm, for the remainder of this chapter we accept any SAHGA solution that has an objective function value within 0.2% of the optimal solution. Fig. 5.2 shows the computational savings that result from implementing this criterion relative to solving fully to convergence to the optimal solution. The savings range from 0% to 98%, with an average of 43%. Even with this change in criterion, however, SAHGA still requires substantially more computational effort than the SGA for Run 6.

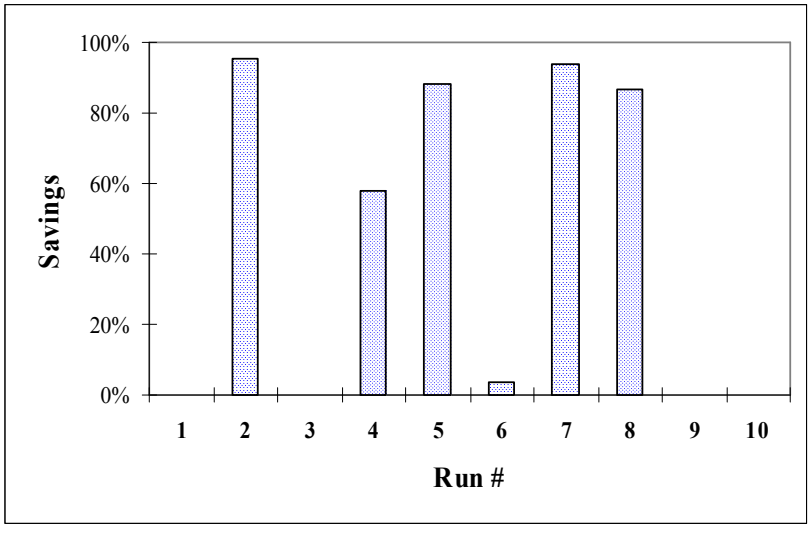



**Fig. 5.2 Savings from full to approximate solution for SAHGA**

The previous results show the necessity to improve SAHGA in order to reduce convergence to sub-optimal solutions and speed up convergence to the optimal solution. The next sections explore algorithmic modifications and local search alternatives for improving performance of the SAHGA algorithm.

## 5.2 Algorithm Reformulation for Improved Performance

This section examines the following potential improvements to the original SAHGA algorithm to reduce computational effort:

- Stopping Criterion;
- End of Local Search;
- Evolution of Best Individual; and
- Individual Selection.

In Section 5.2.1 below, each of these enhancements is described and evaluated on the test case presented in Chapter 4 for the same 10 initial populations used to evaluate the algorithm in section 5.1. Then, combinations of the most promising enhancements are described and tested in Section 5.2.2.

### *5.2.1 SAHGA Enhancements*

**a) Stopping Criterion (SC).** The first enhancement is related to the stopping criterion for the number of local search iterations to be performed in the local search step. The stopping criterion was presented in Eq. (3.3) (Section 3.2.3.c), and it is defined by:

$$\frac{\Delta \text{Global}}{\text{pop}} < \frac{\Delta \text{Local}}{\text{fev}} \tag{5.1}$$

where ΔGlobal is the difference in fitness between the best members of the current and previous generations. This definition only considers improvement in the best individual, but does not represent the population as a whole. In later generations, the best individual often remains unchanged, which means ΔGlobal is equal to zero and local search continues even when the change in fitness by local search is minimal. For this enhancement, ΔGlobal is redefined to be the difference in average fitness between generations, which considers the performance of the entire population. In this way, the algorithm reduces the total number of function evaluations, as well as the possibility of premature convergence to a sub-optimal solution.

The results of the performance analysis for this enhancement are presented in Fig. 5.3. The figure includes the savings for each of the 10 runs (relative to the SGA), represented by the height of the bar, as well as the overall average, 5%, and 95% confidence limits across all 10 runs.. The confidence limits were evaluated using a T distribution (see Hogg and Craig, 1978 for details). In order to be conservative, only the 5% bound and the average will be considered for identifying the best enhancement. For this enhancement, the 5% bound is 56.8% (dashed line), the average is 69.6%, and the 95% bound is 82.4% (solid line). This shows that this enhancement is consistent because the difference between the average (or expected) value and the lower bound is small. Moreover, the savings achieved by this enhancement are always positive, indicating that the enhancement may be overcoming the premature convergence problem presented by the original SAHGA.

**b) End of Local Search (ELS).** After careful examination of results presented in Section 5.1, it became clear that when the difference between the average fitness and the best

fitness in the population is small, local search no longer helps to solve the problem and simply costs additional computational time. For this reason, the second proposed modification to SAHGA is to halt local search when this difference is small. The analysis was performed for three different stopping levels for the relative difference between average and best fitness (ELS parameters): 1%, 0.1% and 0.01%. The results of this analysis, presented in Fig. 5.4, show that when the parameter is set to 0.1%, the simulation is better or equal to the other two simulations for 8 of the 10 runs. This result shows that for this parameter value the algorithm has a good balance, usually stopping the simulation neither too early nor too late. Fig. 5.5 shows a comparison between the original SAHGA algorithm and the best results for the end of local search (0.1 %), again with savings measured relative to the SGA. These results show that the average savings for the enhanced algorithm are 13.6%, the 5% bound is -68.4% and the 95% bound is 82%. Although this enhancement does not always overcome the negative savings associated with the original SAHGA algorithm in some of the runs, it does consistently improve the savings across all runs.

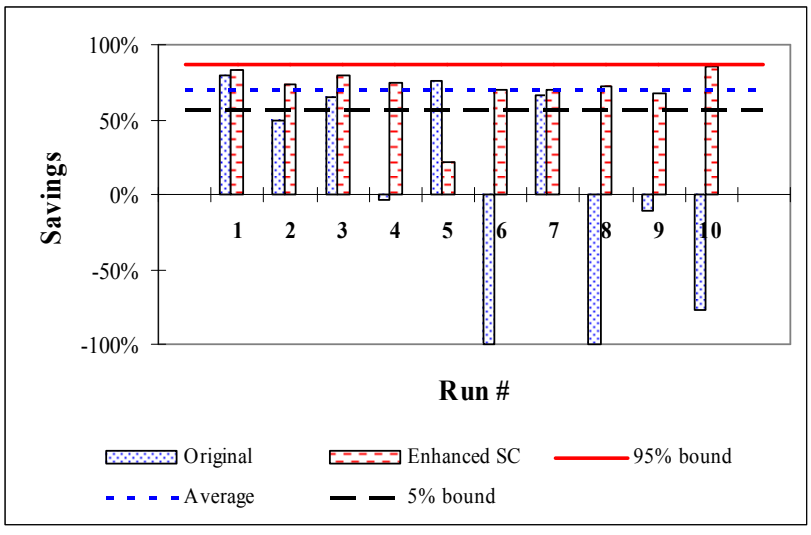



**Fig. 5.3 Results for stopping criterion enhancement**

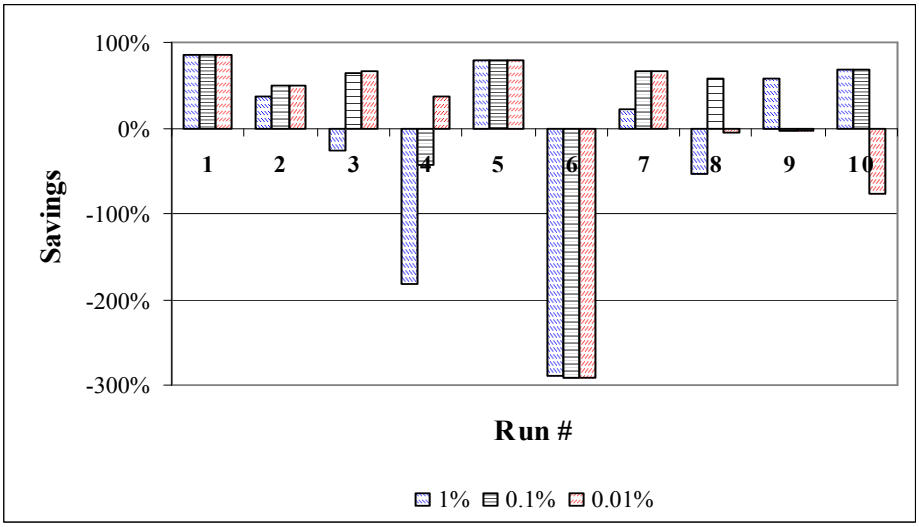



**Fig. 5.4 End of local search results for different ELS parameters**

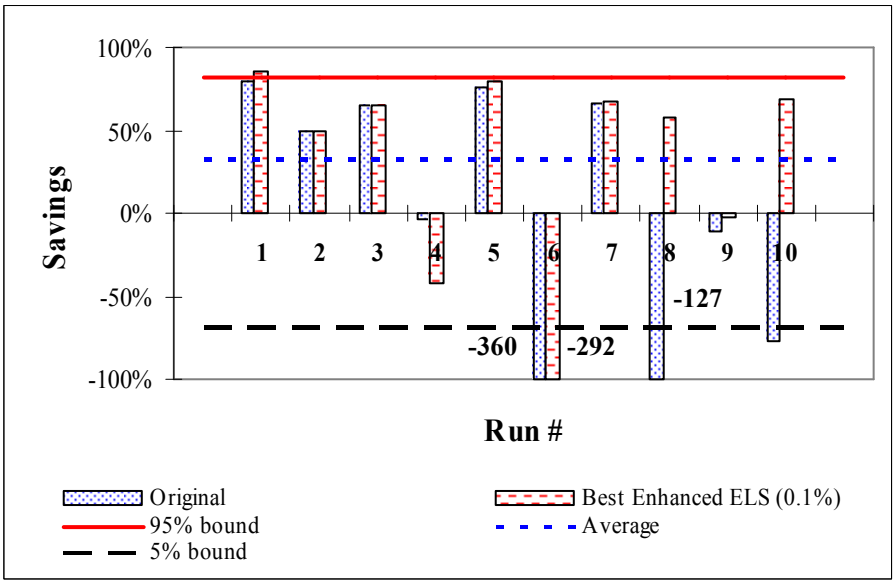



**Fig. 5.5 Performance of best end of local search enhancement**

**c) Evolution of the Best Individual (EBI).** As described in Section 3.2.1, the link between local search and the GA is performed by means of Lamarckian or Baldwinian evolution. Recall that with Lamarckian evolution, the fitness and the chromosome of the improved

individual are both passed from the local search step to the GA. On the other hand, Baldwinian evolution only passes the fitness of the new individual to the GA. Analysis of preliminary trial runs showed that sometimes the best individual does not take over the population. This happens because the individual was modified in the local search step and only its fitness was transferred using Baldwinian evolution. To avoid this problem, after the local search step is complete, the best individual will always be transferred using Lamarckian evolution. Fig. 5.6 shows the performance of this enhancement, with average savings relative to the SGA of 52.2%, the 5% bound equal to 23.1% and the 95% bound of 81.3%.

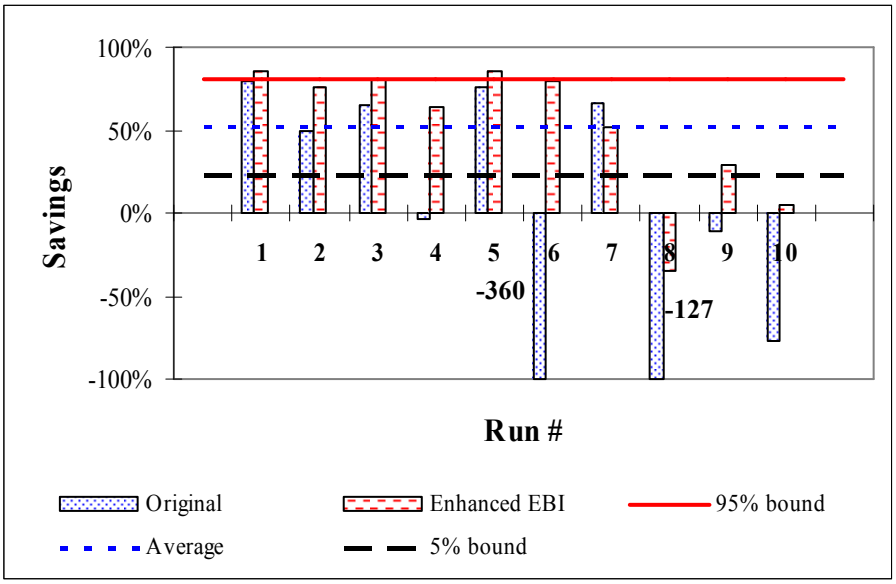



**Fig. 5.6 Results for evolution of best individual enhancement**

**d) Individual Selection (IS).** The fourth and last enhancement is related to the selection of individuals in the population for local search. The current mechanism involves random selection of *N* individuals from the full population. In the new approach, the selection will be performed by means of Latin-Hypercube sampling (Iman et al., 1980) from clusters of

individuals. Clustering ensures that the individuals chosen represent a broad sampling of the population. Four clustering approaches are considered, which are described below.

- **Selection and Order (SO)**: In the first approach, the individuals are first ordered using three different criteria: fitness, total pumping rate, and the two previous elements combined. For the first criterion, the individuals will simply be ordered from the lowest to the highest fitness. Then the population will be divided into *N* groups with equal number of individuals (or as close to equal as possible). Fig. 5.7 shows an example of the clustering process for this criterion with a population of 25 individuals from which three individuals are to be chosen for local search. The individuals are ordered by fitness (F) and then grouped into three clusters. The second criteria will be similar to the first, but using total pumping (Q) rate instead of fitness to perform the ordering. The third criterion will involve clustering (C) the population into *N* clusters with similar fitness and total pumping rates. The clusters were created using K-means clustering (MacQueen, 1967). Fig. 5.8 shows the same example as in Fig. 5.7, but with clusters formed by both fitness and pumping rate simultaneously. K-means clustering takes some computational time (on the order of fraction of seconds for the groundwater remediation test case), but relative to the time required for fitness function evaluations (minutes for the groundwater remediation test case) the time is insignificant.

  Once the clusters are formed using one of these approaches, one individual is selected for local search from every one of the regions presented in Fig. 5.7 (A, B, and C) or every cluster presented in Fig. 5.8 (1, 2, and 3). To choose the individual, three selection

mechanisms are investigated: best (B), random (R) and worst (W). In the first mechanism, the best individual (smallest fitness) in the region or cluster is selected. In the second mechanism, a random individual is selected. Finally, in the last selection process, the worst individual (highest fitness) is selected.

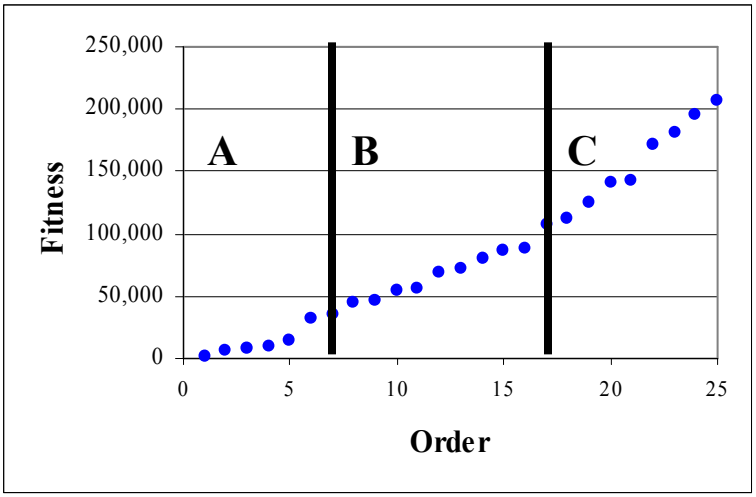



**Fig. 5.7 Clustering by fitness**

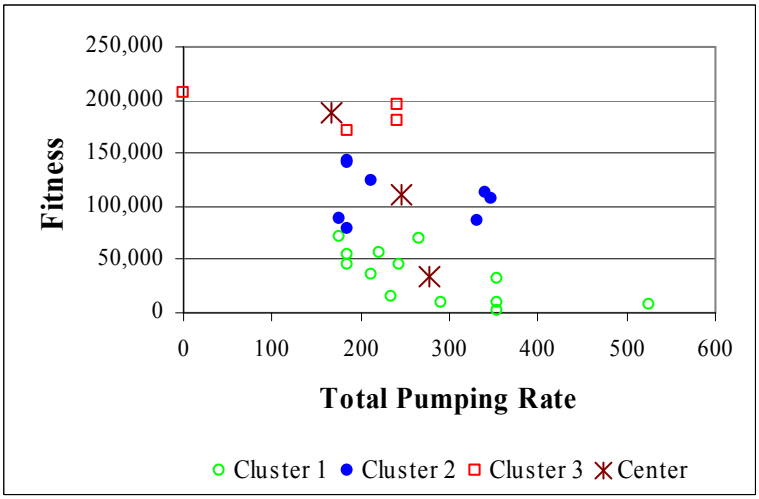



**Fig. 5.8 Clustering by fitness and total pumping rate**

- **Selection, Order and Best Individual (SO&B)**: In the second clustering approach, a total of N+1 individuals are selected to undergo local search, N of them using the previous selection and order approaches, plus the best individual (B) of the previous generation. When the best individual was previously selected (when selecting randomly or the best individual in the cluster), the second best individual will take its place.

- **Feasible or Infeasible (F/I).** For constrained problems, it may be important to ensure that local search is done on both infeasible individuals, which violate all or some of the constraints defining the problem, and feasible individuals, which have no violations of the constraints. For this approach, the population is first divided between $M_F$ feasible individuals and $M_I$ infeasible individuals. The number of individuals to select for local search (N) is then allocated between both groups by selecting $N_I$ infeasible individuals and $N_F$ feasible individuals, calculated proportional to the number of individuals of that class in the total population as shown in Eq. (5.2):

$$N_I = \left\langle \frac{M_I}{M_I + M_F} N \right\rangle, \qquad N_F = N - N_I \tag{5.2}$$
$$\text{where : } \langle x \rangle \text{ represents the nearest integer to x}$$

Fig. 5.9 illustrates this calculation for an example with a total population of 100 individuals and a sample size of 20. The plot shows the number of samples to take from the infeasible and feasible part of the population as a function of the number of infeasible individuals. For example, when the number of infeasible individuals is 40, 8 individuals are selected randomly from the infeasible individuals and 12 from the feasible ones.

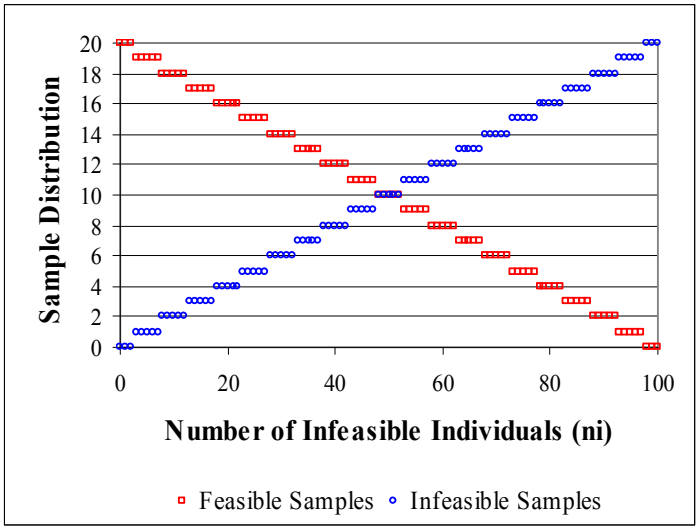



**Fig. 5.9 Sample distribution between feasible and infeasible individuals**

- **Feasible or Infeasible and Best Individual (F/I&B).** In this case, the sample has N+1 individuals: the best individual in the population plus N individuals selected using the previous feasible or infeasible approach.

Using the four different approaches described in the previous paragraphs, a total of 20 different individual selection combinations were created to consider different ways to select and/or order the individuals undergoing local search (see Table 5.1). From the 20 combinations, 16 of them improved the performance of the algorithm for the 10 runs under consideration. Fig. 5.10 shows the best results for clustering selection and order, which occurred using random fitness RF and RF&B, plus the results for the clustering F/I and F/I&B. These results show that on average the best performance is achieved for RF&B, which has clustering by fitness, a random selection mechanism, and the best individual always included. From the figure it is also clear that the worst performance was achieved by the enhancement F/I, with somewhat improved

results for the enhancement F/I&B. The enhancement RF&B is the only one that always gives positive savings relative to the SGA and it is the best enhancement for 4 of the 10 runs.

**Table 5.1 Individual Selection Combinations for Enhancements Analysis**

| Combination | Clustering Criteria | Mechanism for Selecting Individuals from the Cluster |
|---|---|---|
| **BF** | Fitness | Best |
| **RF** | Fitness | Random |
| **WF** | Fitness | Worst |
| **BF&B** | Fitness | Best & Best Individual |
| **RF&B** | Fitness | Random & Best Individual |
| **WF&B** | Fitness | Worst & Best Individual |
| **BQ** | Total Pumping Rate | Best |
| **RQ** | Total Pumping Rate | Random |
| **WQ** | Total Pumping Rate | Worst |
| **BQ&B** | Total Pumping Rate | Best & Best Individual |
| **RQ&B** | Total Pumping Rate | Random & Best Individual |
| **WQ&B** | Total Pumping Rate | Worst & Best Individual |
| **BC** | Fitness & Total Pumping Rate | Best |
| **RC** | Fitness & Total Pumping Rate | Random |
| **WC** | Fitness & Total Pumping Rate | Worst |
| **BC&B** | Fitness & Total Pumping Rate | Best & Best Individual |
| **RC&B** | Fitness & Total Pumping Rate | Random & Best Individual |
| **WC&B** | Fitness & Total Pumping Rate | Worst & Best Individual |
| **F/I** | Feasible/Infeasible | Random |
| **F/I&B** | Feasible/Infeasible | Random & Best Individual |

Fig. 5.11 shows the comparison between the original SAHGA algorithm and the best enhanced (RF&B) approach for individual selection. For this enhancement, the average savings are 60.1%, the lower bound is 37.7%, and the upper bound is 82.4%. The enhanced approach is 4% to 37% slower than the original SAHGA in three runs, but is substantially more reliable in attaining positive savings relative to the SGA. It is important to point out that even when the ordering mechanisms include the use of problem-dependent information (such as total pumping rate), in the end the best approach uses only the fitness of the individual, a parameter valid for all types of applications.

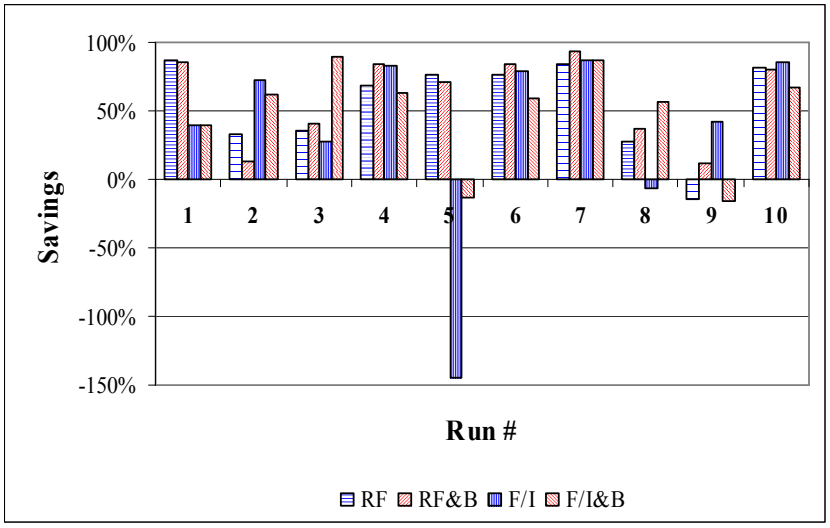



**Fig. 5.10 Results for individual selection:**
**Best results for each clustering**

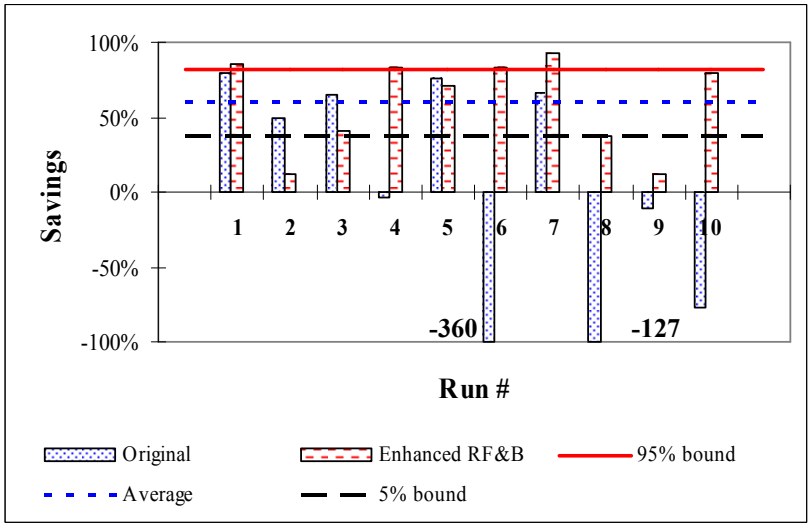



**Fig. 5.11 Results for individual selection enhancement: RF&B**

### *5.2.2 Multiple Enhancements*

In this section, the best performing enhancements from the previous section (end of local search [0.1%], evolution of best individual, random selection with fitness order, and selection of best individual) are tested in the following two combinations:

- **Combination 1**: Stopping Criterion + End of Local Search + Random Selection with Fitness Order & Best Individual
- **Combination 2**: Stopping Criterion + End of Local Search + Random Selection with Fitness Order & Best Individual + Evolution of Best Individual

These four enhancements were combined in this way because of their different nature. The enhancements stopping criterion (SC), end of local search (ELS), and individual selection (IS) act over the search itself, at the beginning (IS) or during the search (SC and ELS), but the final enhancement, evolution of the best individual (EBI) acts after a local search iteration is completed. For this reason, the first three enhancements are combined in the multiple enhancement "Combination 1", and then the last enhancement is added to define the multiple enhancement "Combination 2".

The results for these combinations are presented in Fig. 5.12. The results show that Combination 1 performs substantially better than Combination 2, primarily because Combination 2 induces premature convergence to a sub-optimal solution different from the best known solution for at least 2 realizations of the initial population. For this reason, the algorithm needs to apply the injection approach more extensively. In fact, for combination 1, all the simulations but one found the solution for K=1 and only one of them needed to solve the problem for K=2. On the other hand, for combination 2, it was necessary to apply the injection method for three runs to K=2 and for one run to K=4.

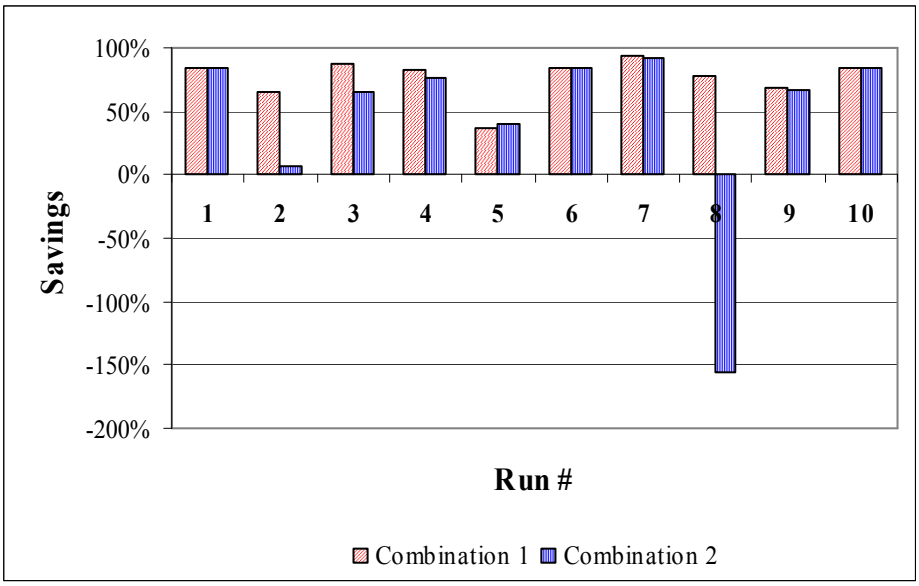



**Fig. 5.12 Multiple enhancement results**

In order to visualize the enhancements in perspective, Fig. 5.13 presents the average savings versus speed up for the best set of enhancements selected in Section 5.2.2. In this figure, the definitions of savings and speed up are:

$$\text{Savings} = 1 - \frac{\langle \text{SAHGA} \rangle}{\langle \text{SGA} \rangle}$$
$$\text{Speed Up} = \frac{\langle \text{SGA} \rangle}{\langle \text{SAHGA} \rangle} - 1 \tag{5.3}$$

Eq. (5.3) shows that when the average number of function evaluations necessary to solve SAHGA is equal to the average number of function evaluations to solve SGA, the savings are zero and the speed up is equal to zero.

Fig. 5.13 shows clearly the average effect of every individual enhancement and the total improvement achieved by the multiple enhancements defined by Combinations 1 and 2. It is interesting to observe the relative position of the four single enhancements in the plot. For

Combination 1 it is clear that the three single enhancements work together to achieve a better result. On the other hand, when the enhancement for evolution of the best individual (EBI) is included in Combination 2 the final effect is to have a performance that is even worse than the performance of the EBI enhancement by itself. This effect occurs because EBI causes premature convergence at lower population sizes when used in combination with the other enhancements.

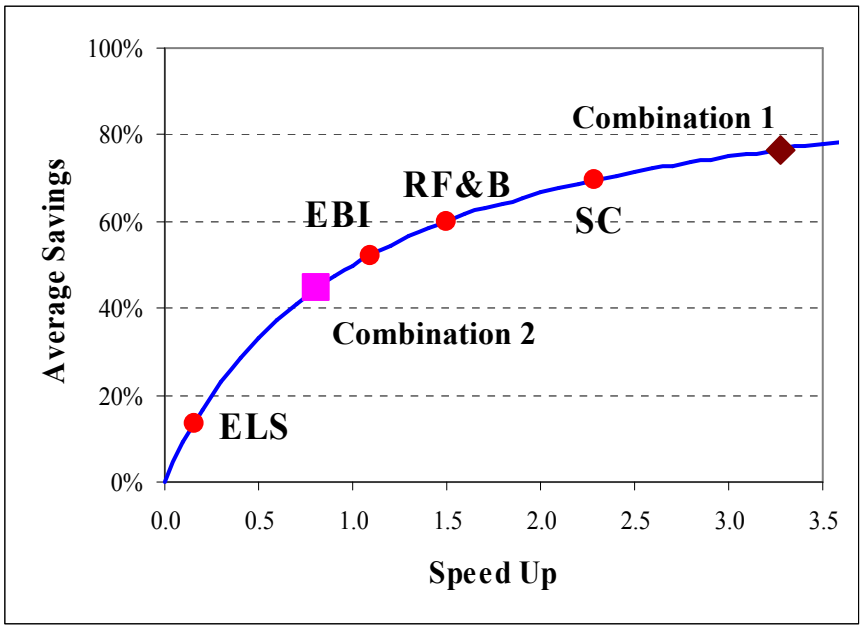



**Fig. 5.13 Visualization of enhancements results**

Finally, Fig. 5.14 shows the comparison between the original algorithm and the best final enhanced algorithm (Combination 1). On average, the enhanced algorithm improves performance roughly 100%, from –24.2% to 76.6%. The 5% bound is 64.9 % and the 95% bound is 88.4%. In essence, this result shows that proper selection of the individuals undergoing local search is essential to achieving good performance, together with a stopping criterion that takes into consideration the behavior of the population undergoing local search as a whole. These

findings will be validated in the final section of this chapter, where the enhanced algorithm will be tested with 100 random starting populations.

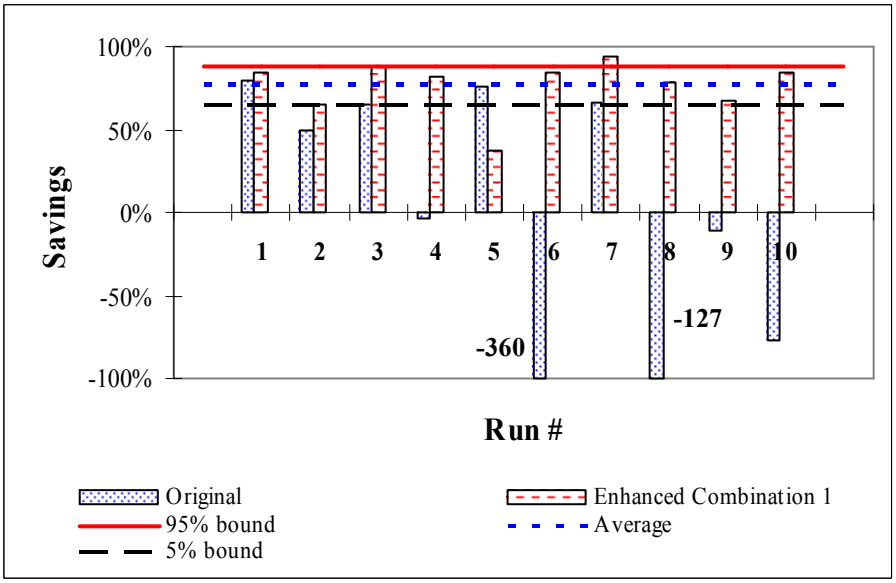



**Fig. 5.14 Results for best multiple enhancement: Combination 1**

## 5.3 Performance of Different Local Search Algorithms

Section 5.2 presented different enhancements to the SAHGA operations to improve performance. This section examines the effects of the local search algorithm by comparing performance of the original SAHGA and the enhanced SAHGA algorithm selected in the previous section (Combination 1) using the five local search algorithms presented in Section 3.3. First, Section 5.3.1 presents the results of the application of the original SAHGA algorithm and the enhanced SAHGA for each local search algorithm. Then, Section 5.3.2 presents further analyses to explain the differences in performance among different local search algorithms from the original to the enhanced SAHGA algorithm.

### *5.3.1 Results for Different Local Search Algorithms*

The analysis was performed for the following 5 local search algortithms (described in Section 3.4):

- Random Walk with Uniform Distribution (LS1)
- Random Walk with Normal Distribution (LS2)
- (1+1)-Evolutionary Strategy (LS3)
- Random Derivative (LS4)
- Steepest Descent (LS5)

For each local search algorithm, appropriate population sizes were identified following the recommendations presented in Section 3.5 (the same for the original and the enhanced SAHGA). The standard deviation reduction (β) was evaluated as the average of the reduction in the first 5 iterations, shown in Fig. 5.15. Table 5.2 shows the value of β for all the algorithms and the corresponding population size for K=1. Table 5.3 shows the population sizes for all values of K for each local search algorithm. As in Section 5.1, each algorithm was run for increasing values of K until the optimal solution is within 0.2% of the best known solution to the problem.

`Fig. 5.16 shows the average savings relative to the SGA for the original and the enhanced algorithm with each local search algorithm. The results for the enhanced algorithm show clear improvement over the original algorithm for 4 of the 5 local search approaches. Only LS5 is slower, by an average of 3%. The best results for the enhanced algorithm are attained by random uniform local search (LS1), followed by random normal local search (LS2). Fig. 5.17 shows the detailed results of the 10 runs for each of the local search algorithms with the

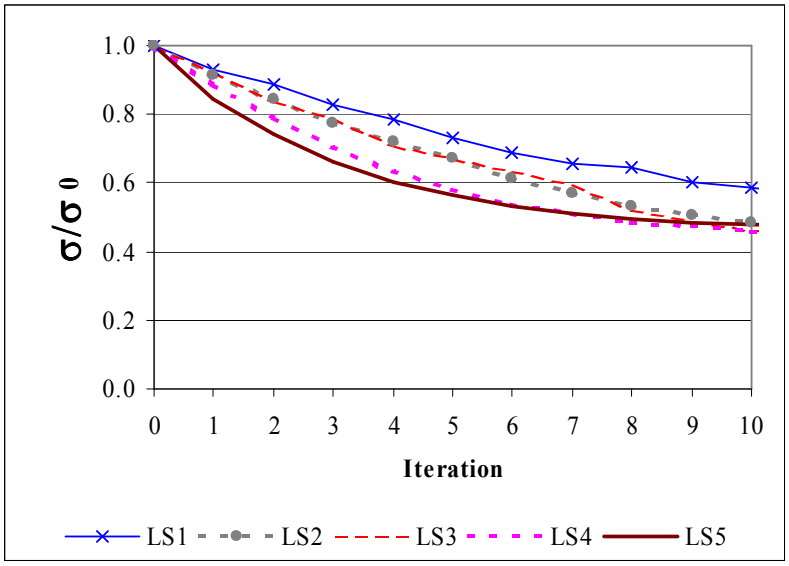



**Fig. 5.15 Standard deviation reduction for each local search algorithm**

**Table 5.2 Base Population Size for Each Local Search Algorithm**

| LS | β | SAHGA Population (K=1) |
|---|---|---|
| **1** | 0.83 | 25 |
| **2** | 0.78 | 24 |
| **3** | 0.78 | 24 |
| **4** | 0.72 | 22 |
| **5** | 0.68 | 20 |

**Table 5.3 SAHGA Population Sizes for Each Local Search Algorithm**

| | | Local Search Algorithm | | | | |
|---|---|---|---|---|---|---|
| **K** | **SGA** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **1** | 30 | 25 | 24 | 24 | 22 | 20 |
| **2** | 60 | 50 | 48 | 48 | 44 | 40 |
| **3** | 120 | 100 | 96 | 96 | 88 | 80 |
| **4** | 240 | 200 | 192 | 192 | 176 | 160 |
| **5** | 480 | 400 | 384 | 384 | 352 | 320 |

enhanced algorithm. From Fig. 5.17, it is clear that LS1 and LS2 have much more reliable performance for the enhanced SAHGA relative to the SGA than the other local search algorithms. LS1 and LS2 are similar algorithms, but LS1 explores the entire local neighborhood around the current solution with the same probability while LS2 explores the area closer to the current solution with higher probability. The higher likelihood of a broader search in LS1 apparently improves performance for this problem. For (1+1)-ES (LS3), the runs with poor results appear to be related to premature convergence to a sub-optimal solution. Finally, random gradient (LS4) and gradient (LS5) sometimes have poor results because of the additional computational effort required for these algorithms.

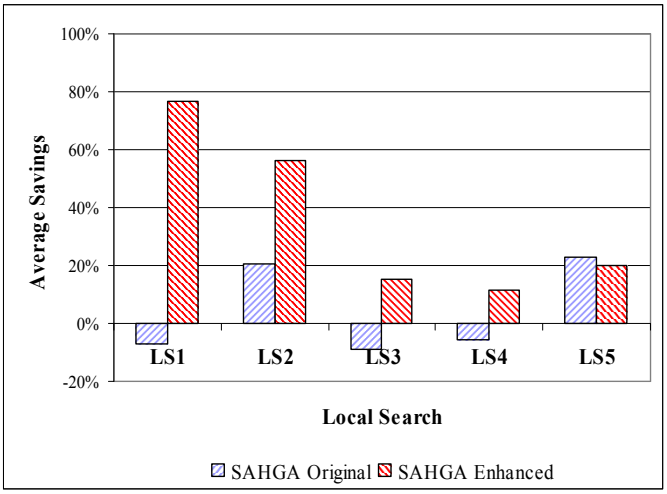



**Fig. 5.16 Results for different local search algorithms**

### *5.3.2 Analysis of Enhanced SAHGA Effectivity*

The results presented in the previous section show that random local search algorithms, specifically LS1 and LS2, perform best with the enhanced SAHGA algorithm but the gradient-

based approach (LS5) performs best with the original algorithm. To explore the reasons for this change, the following two measures are created that show the effectivity of the new algorithm:

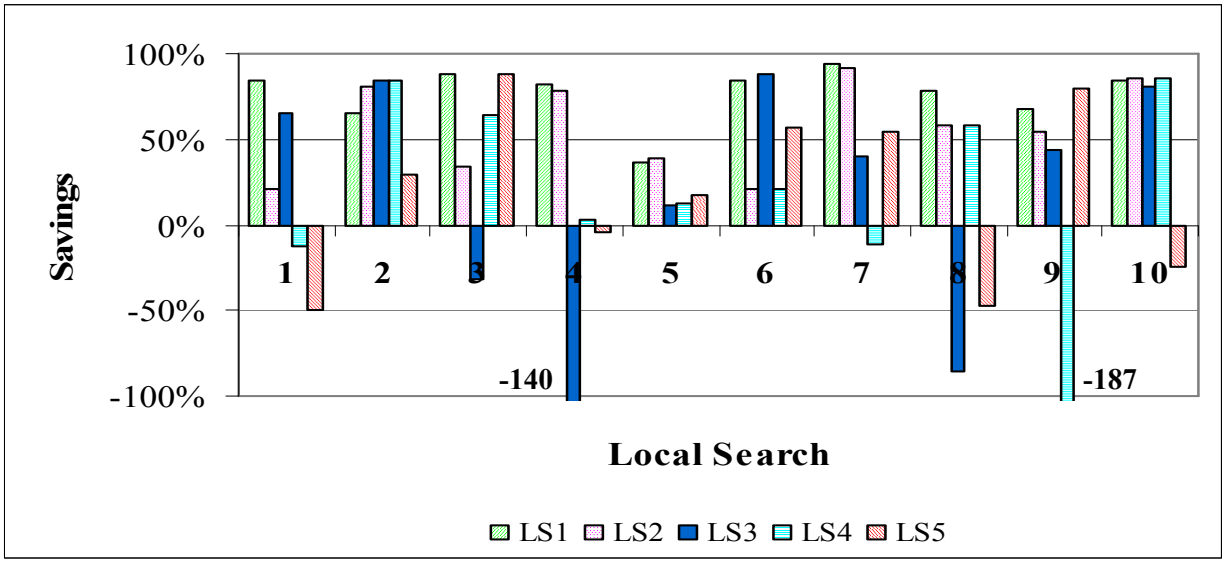



**Fig. 5.17 Detailed results for each local search algorithm using enhanced SAHGA**

1. **Local Search Effectivity:** Ratio of local search callings that are effective in improving the solution

$$Effectivity = 100\,\frac{Effective\ \ Local\ Search\ Calllings}{Total\ Local\ Search\ Calllings}$$

2. **Difference in LS Calls:** Difference in number of local search calls between SAHGA and Enhanced SAHGA.

$$Difference = 100\,\frac{Total\ LS\ calls\ Enhanced\ SAHGA - Total\ LS\ calls\ SAHGA}{Total\ LS\ calls\ SAHGA}$$

The first measure shows the effectiveness of the local search step for both algorithms. On the other hand, the second measure evaluates how different the algorithms are with respect to the total number of times local search is used.

Fig. 5.18 and 5.19 show the average results for the first and second measure, respectively, across the same 10 runs discussed previously. It is clear that the enhanced SAHGA algorithm improves both measures for LS1, LS2, and LS3 more than for LS4 and LS5. This result most likely occurs because the enhancements to SAHGA primarily improve the selection of individuals for local search. This type of enhancement is more important for random local search algorithms such as LS1, LS2, and LS3 because gradient-based local searchers are more effective at finding improved solutions from any starting point. For this reason there is virtually no improvement in local search effectivity (Fig. 5.18) for LS4 and LS5 with the enhancements. The second measure, presented in Fig. 5.19, shows that the number of local search calls decreases considerably for LS1, LS2, LS3, and LS4 also decrease somewhat, and LS5 increases somewhat. Overall, these findings illustrate that the change in local search algorithm performance results from a combination of improved local search effectivity and a reduction in local search calls, causing LS1 to replace LS5 as the best-performing local search algorithm.

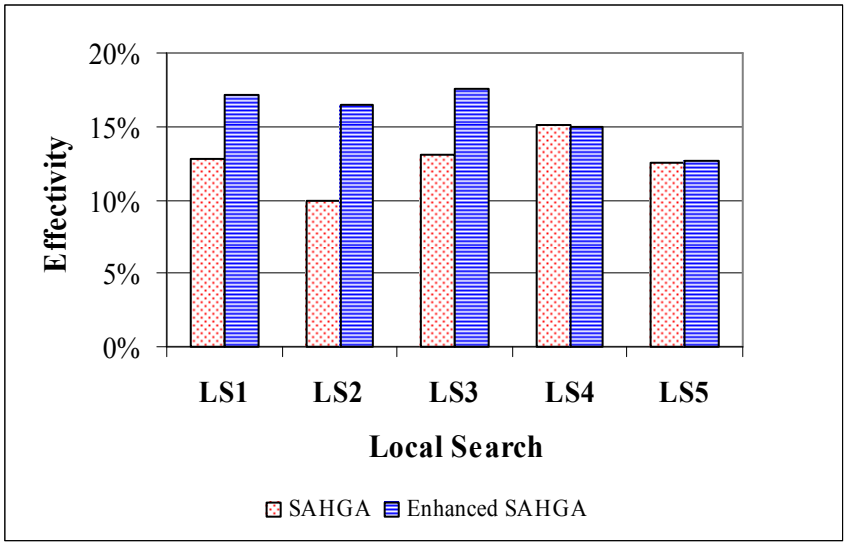



**Fig. 5.18 Local search effectivity**

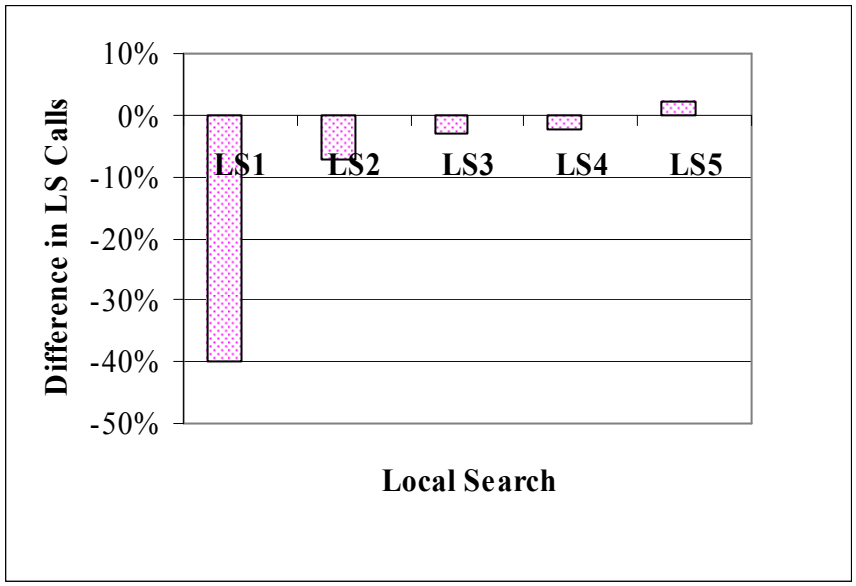



**Fig. 5.19 Difference in local search calls from SAHGA to enhanced SAHGA**

To further verify that the cause of the change in local search performance is a result of the algorithmic changes, the local search results for the two algorithms are next verified with a larger population size. This analysis was necessary because more random local search algorithms could also enhance performance when the population size is too small, bringing more random search to a small population that could otherwise have insufficient random search capabilities. To verify that the local search algorithms' performance is the same for larger population sizes, a population of 240 individuals (K=4) was selected for the SGA (see Table 4.3), and with this base value, the population for the enhanced SAHGA was evaluated as 200, 192, 192, 176, 160 for LS1, LS2, LS2, LS4, and LS5, respectively.

The results of the analysis are presented in Fig. 5.20 and 5.21. Fig. 5.20 shows the average savings for both SAHGA algorithms relative to the SGA, using each of the five local search algorithms. Comparing this graph to Fig. 5.16, which used smaller population sizes with

the injection approach, the general trends in the results are similar. However, the performance of the enhanced algorithm is now better for all of the five different local searches, including LS5. At the same time, for the enhanced algorithm the average number of function evaluations is similar for all local search algorithms. Fig.5.21 also shows that the largest improvements between SAHGA and enhanced SAHGA are achieved for LS1 and the smallest for LS5. These results are similar to the results attained by the injection approach.

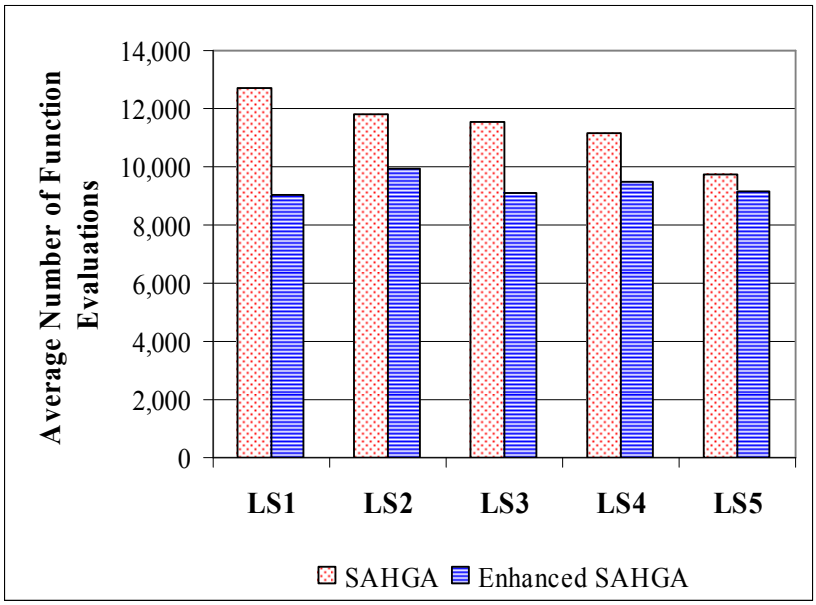



**Fig. 5.20 Savings relative to SGA for SAHGA and enhanced SAHGA with larger population size**

A final test was done to determine whether the relative performance of the different local search algorithms under the original and enhanced algorithms also changes for the eight test functions studied in Chapter 3. Appendix B presents the results of this analysis, which show the same general trend. The largest improvements between SAHGA and the enhanced SAHGA are again achieved for the random search algorithms (usually LS1), but the overall best performing

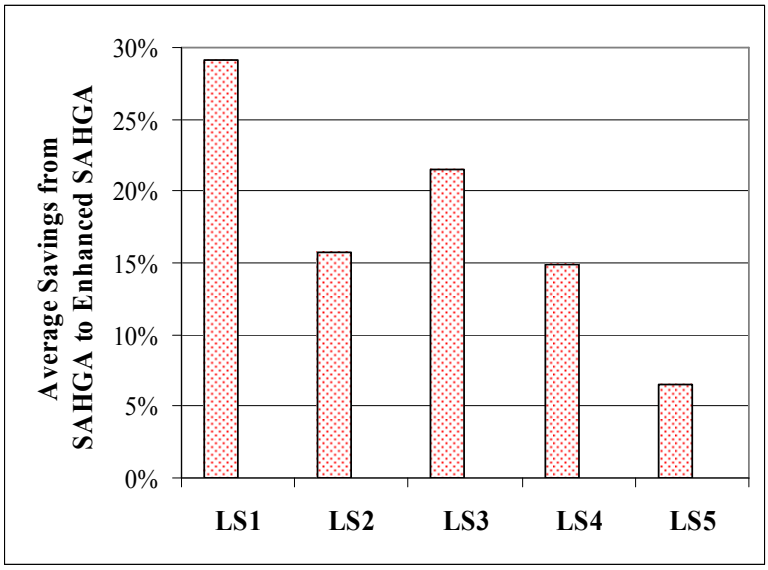



**Fig. 5.21 Relative difference between SAHGA algorithms for larger population size**

local search algorithm for each function can again be related to the difference in population sizes. When the difference in population size among the local search algorithms is significant (see Table 3.5), it is more likely that the gradient-based algorithms (LS4 or LS5), which allow smaller population sizes, will have better performance. On the other hand, when the difference in population sizes is less significant, it is less likely that the gradient-based algorithms will be worth the additional function evaluations. For the groundwater remediation problem, Table 5.3 shows that the population size for LS5 is 20 (K=1), 80% of the population size for LS1 (LS5/LS1=80%). This small reduction in population size can not compete against the extra effort associated with the use of LS5, which explains the improved performance for LS1. In contrast, for the test problems where LS5 is better than LS1, the ratio of population sizes (LS5/LS1) is less than 60%.

## 5.4 Algorithm Validation

In this section, the best-performing enhanced algorithm from the previous section (Combination 1 with LS1) is tested more extensively to validate the previous results. First, the original and the enhanced algorithm are evaluated by running 100 realizations of the initial population. Fig. 5.22 shows the results of the simulations, which are given in terms of the probability distributions of the computational savings of both algorithms relative to the SGA. It is clear that the original algorithm is slower than the enhanced one because the savings for the original algorithm are as low as -500%. For the original algorithm, savings are negative (i.e., the SGA is faster) 20% of the time and the average savings are equal to 14%. On the other hand, for the enhanced algorithm, the savings are negative only 10% of the time and the average savings are equal to 64%. The negative savings occur because the enhanced SAHGA is designed to explore the decision space more carefully than the SGA, a capability that is not needed when the initial population contains much of the final solution and costs more computational effort. Overall, however, the enhanced SAHGA algorithm is generally much faster and more reliable than either the original SAHGA or the SGA. Fig. 5.23 shows the reliability of the algorithm as a function of the population sizes corresponding to each building block order, where reliability is defined as the percentage of runs that came within 0.2% of the optimal solution at that population size. From the plot it is clear that 90% of the enhanced SAHGA runs came within 0.2% of the optimal solution for a building block order of 2 (K = 2), but for SGA this reliability is achieved only for K=4 and for the original SAHGA for K=3. These results show that the enhanced SAHGA algorithm is better than both the SGA and the original SAHGA algorithm proposed in Chapter 3.

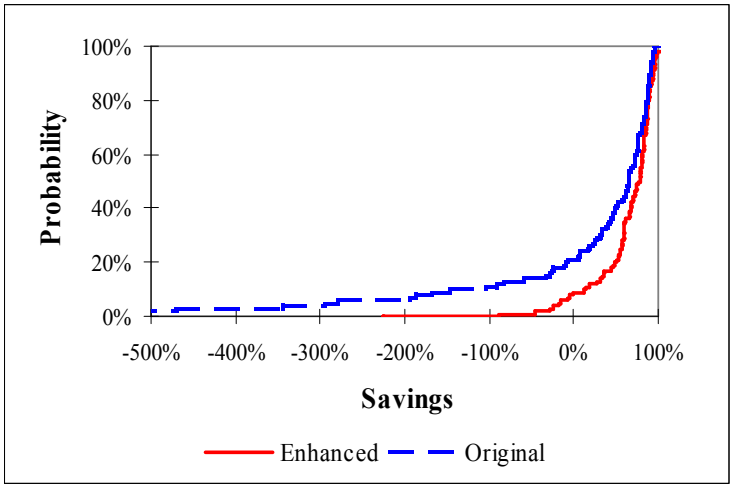



**Fig. 5.22 Probability distribution of computational savings for SAHGA algorithms**

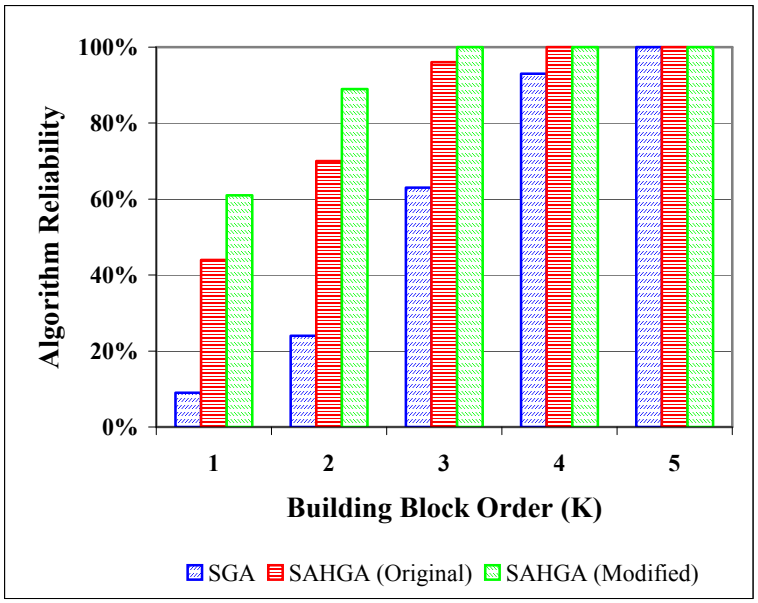



**Fig. 5.23 Algorithm reliability**

## 5.5 Conclusions and Recommendations

The results presented in this chapter showed that the original SAHGA algorithm proposed in Chapter 3 is not as reliable on the groundwater remediation case study as would be expected based on its performance on the test functions in Chapter 3.

To improve performance, a number of enhancements to the original algorithm were proposed and tested. The most effective modifications were a fitness-based clustering approach to select the individuals undergoing local search more effectively and a new mechanism to stop local search when it is no longer needed. A test of these modifications on 100 initial populations found that the average savings relative to the SGA improved from 14% to 64% when the best combination of enhancements were used. Moreover, the enhanced algorithm achieves 90% reliability in solving the problem faster than the SGA, as compared with the original algorithm that is 80% reliable. It is important to note that the final enhancement does not work with problem-specific information; hence, the enhanced SAHGA algorithm, hereafter called e-SAHGA, can be applied to other types of optimization problems.

The results of the local search analysis showed that the best performance was attained for random walk with uniform distribution. This result is surprising given that this algorithm attains the least reduction in standard deviation in Fig. 5.15, which indicates that it had the least improvement in fitness per iteration. The lower computational effort required for each iteration of the algorithm, compared with the derivative-based methods, apparently outweighs the more accurate search capabilities of the derivative-based methods. Relative to the other two random local search methods, random normal and (1+1)-ES, the random search of the neighborhood surrounding the current point appears to improve performance for this problem. The results presented in Appendix B show that this finding also holds for all test functions where the difference in standard deviation reduction for different local search algorithms is not significant. Only when the difference in standard deviation reduction is significant are the more expensive gradient-based local search algorithms sometimes more effective.

# Chapter 6

## APPLICATION OF e-SAHGA ALGORITHM TO REMEDIATION DESIGN AT UMATILLA ARMY DEPOT, OREGON

Previous chapters have tested the performance of SAHGA only on test functions or a hypothetical groundwater remediation problem. This chapter presents the application of the enhanced algorithm (e-SAHGA) to the design of a real problem, the remediation system for Umatilla Army Depot, Oregon. To address the extensive computational requirements of this problem, the chapter also tests the performance of both SAHGA and the SGA in solving the problem using domain decomposition. Section 6.1 includes a description of the Umatilla site together with the mathematical formulation solved in this study. Then, Section 6.2 presents two different approaches to solve the problem: decomposed and full. Section 6.3 presents a comparison of the results with the original design for the system. Finally, Section 6.4 presents the conclusions of this study.

### 6.1 Description

The Umatilla Army Depot is located in northeastern Oregon, in the border between Morrow and Umatilla Counties (see Fig. 6.1). Umatilla Chemical Depot, established in 1941, is an 80 km² military reservation used as an ordnance depot for storage and handling of conventional and chemical munitions (disassembly, modification, reassembly, and repacking). From the 1950s to 1965, about 321 million liters of wash water were discharged into two unlined lagoons at the site. Part of the contaminated wash water migrated to a shallow, unconfined

aquifer located approximately 14 m under the depot surface. The two major contaminants present in this aquifer are *2,4,6-tri-nitro toluene* (TNT) and *hexahydro-1,3,5-trinitro-1,3,5-triazine* (Royal Demolition Explosive or RDX), which are the targets for cleanup in this design. This site was placed on the EPA's National Priorities List (NPL) in 1984 because of the soil and groundwater contamination. Fig. 6.2 shows the existing RDX and TNT plumes as of October 2000, the starting date of an Environmental Security Technology Certification Program (ESTCP) optimization demonstration project (ESTCP, 2003). The unusual RDX plume shape was caused by contaminants stored in different sectors of the site, which migrated over time due to groundwater transport to create the butterfly shape.

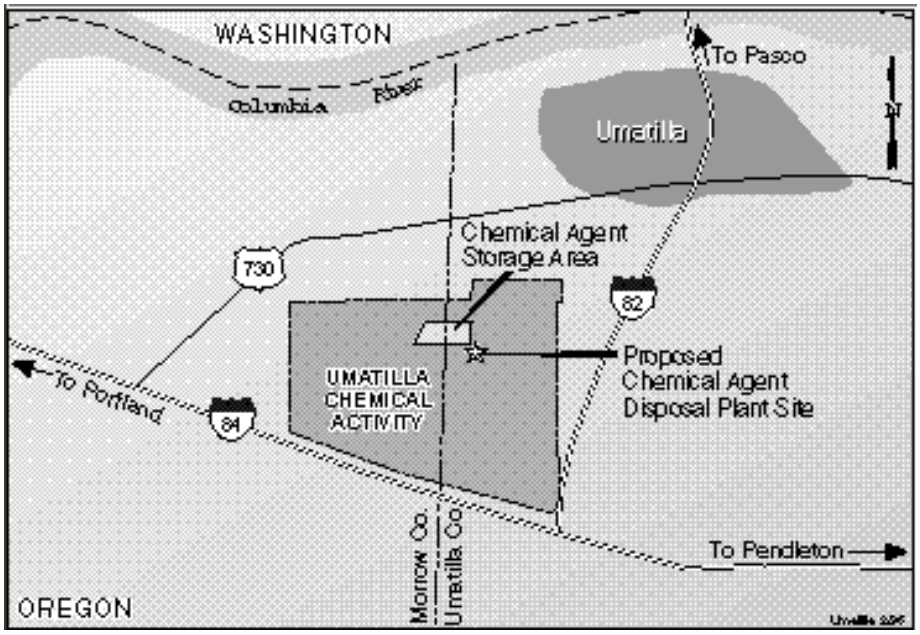



**Fig. 6.1 Umatilla depot location**
**Source: www.globalsecurity.org/wmd/facility/umatilla.htm**

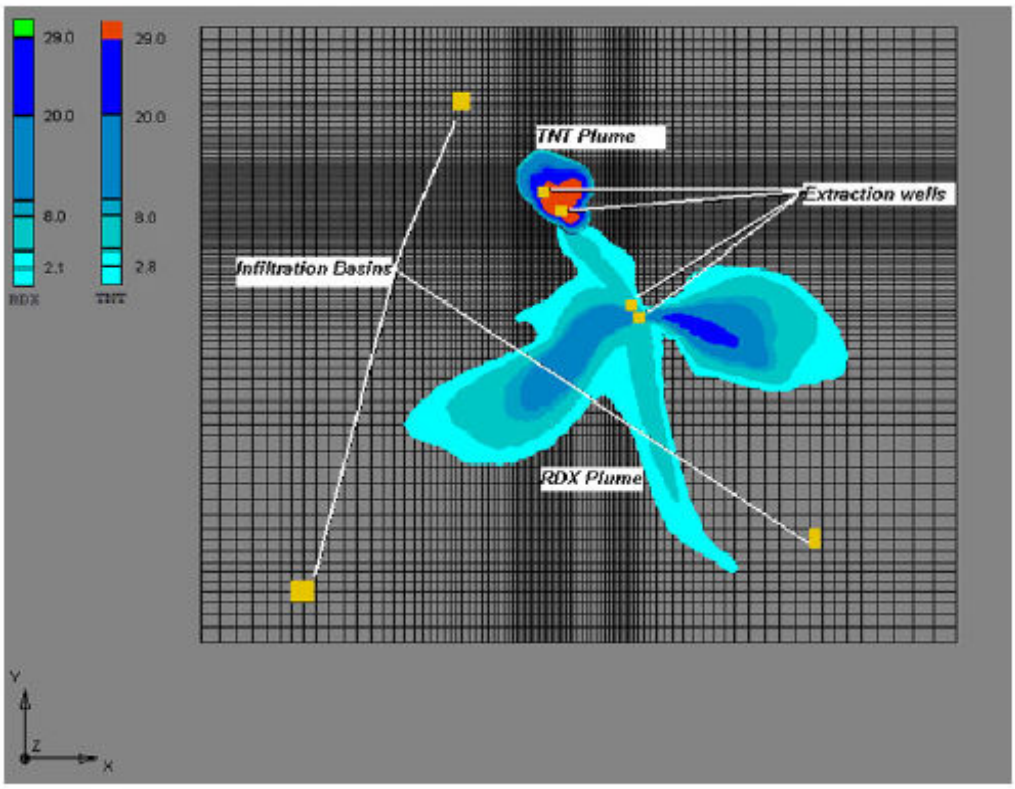



**Fig. 6.2 Existing RDX and TNT plume at the site as of October 2000, along with the pumping wells and infiltration basins**
**Source: ESTCP (2003)**

The U.S. Army Corps of Engineers (USACE, 1996 and 2000) designed a pump-and-treat system to contain and remove the RDX and TNT plumes. The total cost of this design was $ 3.8 million. Fig. 6.2 shows the remediation system, which has three extraction wells and three injection basins to return the extracted water to the aquifer after treatment. The water is treated using granular activated carbon columns. The design of the system was performed by trial-and-error using a simulation model designed to predict the groundwater flow and contaminant transport. The USACE (2000) evaluated flow and contaminant transport at the site using the simulation models MODFLOW 2000 (Harbaugh et al., 2000) and MT3DMS (*Zheng and Wang*, 1999). The entire site was divided into 5 model layers with 132 rows and 125 columns. 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 conductivity ranging from $10^{-6}$ to $10^{-3}$ m/s.

The Umatilla site was one of three demonstration sites chosen for transport optimization of the pump and treat system in the ESTCP study (2003). Working with the site, the ESTCP project developed three optimization formulations. In this research, we focus on the first formulation, which identifies combinations of extraction wells and infiltration basins and associated pumping rates that would minimize cost. The specific conditions for the design are:

- Maximum number of new wells: 4
- Maximum number of new infiltration basins: 3
- Treatment technology for extracted water: granular activated carbon (identical to original design)
- Maximum cleanup time: 5 years
- Maximum concentration allowed in the aquifer by the end of the cleanup: RDX 2.1 μg/L, TNT 2.8 μg/L.

The objective function is presented in Eq. (6.1). Eqs. (6.2) to (6.7) summarize the constraints (ESTCP, 2003):

$$\begin{aligned} &\text{Min} \quad CCW + CCB + CCG + FCL + FCE + VCE + VCG + VCS \\ &\text{st} \end{aligned} \tag{6.1}$$

$$Q_1 + Q_2 \le 1170 \text{ gpm} \tag{6.2}$$

$$Q_1 \le 360 \text{ gpm} \tag{6.3}$$

$$Q_2 \leq 900 \text{ gpm} \tag{6.4}$$

$$C_{RDX} \leq 2.1 \text{ μg/L} \tag{6.5}$$

$$C_{TNT} \leq 2.8 \text{ μg/L} \tag{6.6}$$

$$\text{Total Injection} = \text{Total Extraction} \tag{6.7}$$

where: CCX is the capital cost for the new wells (X=W), new recharge basins (X=B), and new GAC unit (X=G). FCX is the fixed cost for labor (X=L) and electricity (X=E); and VCX is the variable cost for operating the wells (X=E), change of GAC unit (X=G), and for sampling (X=S). $Q_1$ and $Q_2$ are the total pumping rates from two different hydrogeological zones identified in the area of study. More details on the terms in Equations 6.1 through 6.8 can be found in the ESTCP project report (ESTCP, 2003). To solve this problem, the constraints (except for Equations 6.3 and 6.4) are incorporated into the objective function using a penalty function, as described in 4.1.1 for the hypothetical case. This creates a fitness function that is minimized to find the optimal solution.

## 6.2 Solution Approach

To solve the Umatilla case using a GA, the designs are first defined by strings 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. In total, it is then necessary to evaluate 29 decision variables:

- 7 locations of new wells and 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 shown in Fig. 6.2;
- 7 installation flags (new wells and basins);
- 7 installation flags (old wells and basins); and
- 8 pumping rates (for new and old wells), represented by 10 bits for each well.

In terms of binary numbers, these 29 parameters are equivalent to a total of 131 bits. Following the recommendations presented in Section 3.4, the maximum number of generations for this problem is 262 and the base population (K=1) is 40 individuals. Combining population size and number of generations, the expected number of function evaluations for K=1 is approximately 10,000. This is equivalent to 3 days of computation when the evaluation of the objective function takes an average time of one minute (Dell computer with Pentium 4 ESTCP, 2003, 2.4 GH).

The ESTCP project team overcame these computational requirements using several approaches, including domain decomposition, where the solution of the problem is divided into two steps:

- **Step 1:** The optimal locations for wells and infiltration basins are evaluated for the case where the wells are pumping at maximum capacity (360 or 900 gpm). Using this approach, the number of parameters to evaluate decreased to 21.
- **Step 2:** The optimal pumping rates are evaluated for the best design identified in step 1. In this step, the number of parameters to evaluate is 8. It is important to note that this is

the upper bound for the number of pumping rates to evaluate, because pumping rates are evaluated only for wells selected for installation in the first step,.

In this dissertation, we compare the decomposed approach with the full solution approach to evaluate its effectiveness using both SGA and e-SAHGA. Table 6.1 summarizes the full solution approach and the 2 steps for the decomposed approach. This table shows the total number of parameters, the string length, the standard deviation reduction, the population size, the maximum number of generations, and the estimated number of fitness function evaluations required for solution; these values were estimated following the recommendations from Section 4.2. For step 1 of the decomposed approach there is no standard deviation reduction because e-SAHGA is only applied to pumping rates, which are not considered in step 1. The most important result included in this table is related to population size. When the population size decreased from 40 individuals for the full approach to 15 individuals for each of the steps of the decomposed approach, the total number of function evaluations decreased from 10,000 to 3,000. In Table 6.1, it is also important to note the difference between the standard deviation reduction for the full and decomposed approach. For the full approach the reduction is larger because there is more variability in the starting population when the well locations differ among the initial designs.

## 6.3 Analysis of Results

The results attained for the Umatilla system will be presented using the same approaches as in Section 4.3.2: evolution of the best individual and convergence ratio. In all cases, the runs were completed using the sequential injection approach described previously. The run for each

population size was ended when more than 90% of the population converged to the same solution; all runs were halted when two subsequent populations obtained the same solution. Section 6.3.1 presents the results for the domain decomposition approach. Then Section 6.3.2 gives the results for the full approach. Finally, Section 6.3.3 presents the comparison of results between these two approaches. Note that in this analysis, we focus solely on the performance of different solution approaches. The actual pumping solution obtained was the same as that identified by researchers in the ESTCP project (2003) and is not discussed here.

**Table 6.1 Comparison Between Full and Decomposed Solution Approach**

| | | Solution Approach | | |
|---|---|---|---|---|
| | | Domain Decomposition | | Full |
| | | Step 1: Well Locations | Step 2: Pumping Rates | |
| **Total Number of Parameters** | | 21 | 3 | 29 |
| **String Length (l)** | | 51 | 30 | 131 |
| **Maximum Number of Generations** | | 102 | 60 | 262 |
| **Standard Deviation reduction (β)** | | - | 0.80 | 0.75 |
| **Population (K=1)** | **SGA** | 15 | 15 | 40 |
| | **e-SAHGA** | - | 12 | 30 |
| **Estimated Number of Function Evaluations** | **SGA** | 1,600 | 900 | 11,000 |
| | **e-SAHGA (*)** | - | 1,000 | 10,000 |

(*): Includes estimation of the number of function evaluations required in the local search step

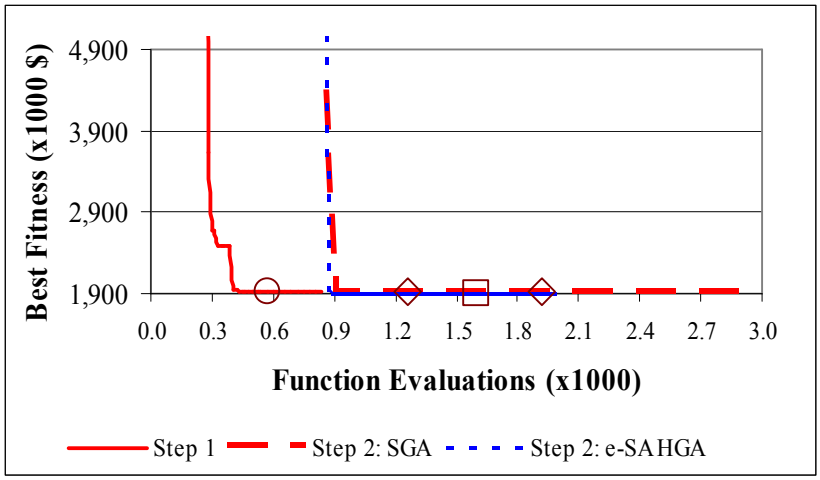
### *6.3.1 Domain Decomposition*

The summary of results of the decomposed approach is presented in Table 6.2. Fig. 6.3 presents the evolution of the best individual and Fig. 6.4 the convergence ratio for steps 1 and 2. In these figures the injection points represent where the population size was increased and the best solution was injected into the new population. For Step 1, the results presented were attained

by applying the algorithm for population sizes K=1 and K=2 because there was no change in fitness between these 2 injection steps. In this step there is no application of e-SAHGA because local search is applied only to pumping rates and in Step 1 the pumping rates are constant. Step 2 is solved using both SGA and e-SAHGA for comparison.

**Table 6.2 Summary of Results for Decomposed Approach**

| | Step 1 | | Step 2 | | | |
|---|---|---|---|---|---|---|
| | SGA | | SGA | | e-SAHGA | |
| **K** | **Final Generation** | **Total Function Evaluations** | **Final Generation** | **Total Number of Function Evaluations** | **Final Generation** | **Total Number of Function Evaluations** |
| 1 | 35 | 540 | 25 | 390 | 59 | 720 |
| 2 | 9 | 300 | 20 | 600 | 16 | 408 |
| 3 | | | 17 | 1,080 | | |
| **Total** | | **840** | | **2,070** | | **1,128** |

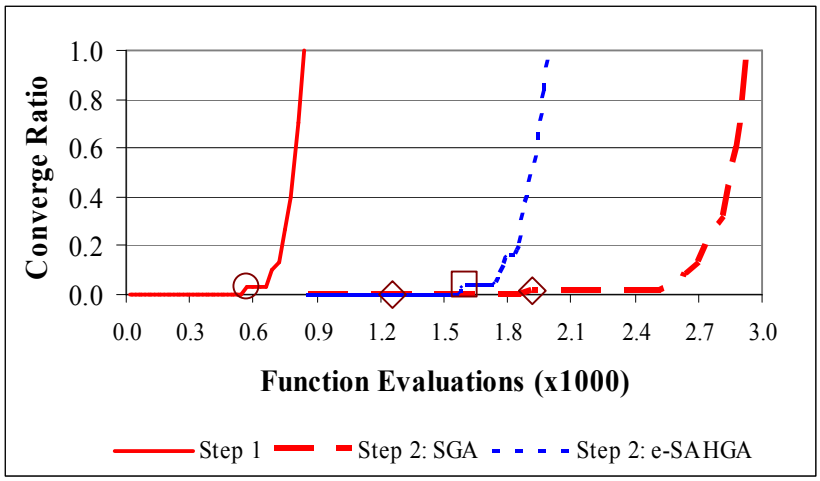



**Injection Points: Step 1: O Step 2-SGA: ◊ Step 2-e-SAHGA: □**

**Fig. 6.3 Domain decomposition approach: Evolution of the best individual**

From the results shown in Table 6.2, it is clear that e-SAHGA is faster than the SGA. For the SGA, it was necessary to solve the problem for population sizes K=1, K=2, and K=3, but for e-SAGHA only sizes K=1 and K=2 were necessary. In total (Steps 1 and 2), the SGA needed

2,910 function evaluations to solve the problem and e-SAHGA only 1,968, a 32.4% savings. Fig. 6.3 shows little difference in the evolution of the best individual in terms of fitness value for the two algorithms, but Fig 6.4 shows clearly that e-SAHGA converges to the optimal solution much faster than the SGA.




**Injection Points:** **Step 1: O** **Step 2-SGA: ◊** **Step 2-e-SAHGA: □**

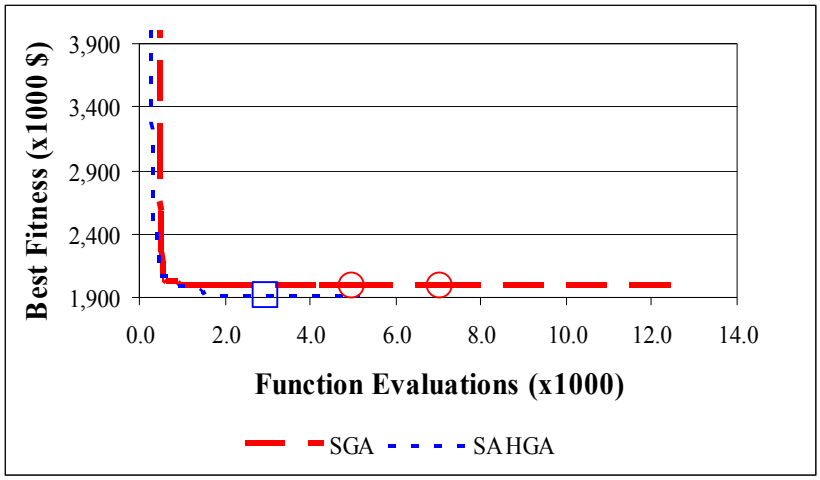
**Fig. 6.4 Domain decomposition approach: Convergence ratio**

### *6.3.2 Full Approach*

Table 6.3 summarizes the results for the full approach. The e-SAHGA algorithm requires 5,169 function evaluations, versus 12,000 for the SGA, a 56.9% reduction. Fig. 6.5, the results for the evolution of the best individual, shows that e-SAHGA identifies a 4% better solution than SGA. Moreover, the convergence ratio in Fig. 6.6 shows substantially faster identification and convergence to the best solution. Note that the SGA was stopped at K=3 because there was no difference between the solutions attained for K=2 and K=3. It is possible that the SGA operated for K=4 or more would identify the best solution that e-SAHGA found, but the computational requirements would increase even more.

**Table 6.3 Summary of Results for Full Approach**

| K | SGA | | e-SAHGA | |
|---|---|---|---|---|
| | Final Generation | Total Function Evaluations | Final Generation | Total Function Evaluations |
| 1 | 121 | 4,880 | 86 | 2,897 |
| 2 | 26 | 2,160 | 35 | 2,272 |
| 3 | 35 | 5,760 | | |
| **Total** | | **12,000** | | **5,169** |



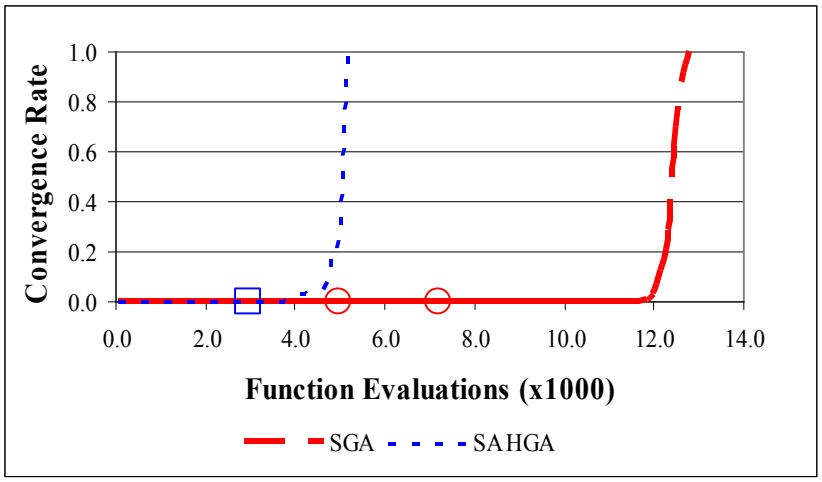

**Injection Points: SGA: O e-SAHGA: □**

**Fig. 6.5 Full approach: Evolution of the best individual**

### *6.3.3 Comparison between Domain Decomposition and Full Approach*

The final step in the analysis is to perform an overall comparison of the domain decomposition and full approaches presented in the previous sections. The results are summarized in Table 6.4, in terms of number of function evaluations and solution quality. The results presented in Table 6.4 show that the savings between domain decomposition and the full approach are more substantial for the SGA algorithm. However, this result is deceptive because the solution by SGA takes longer than the solution by e-SAHGA for both solution approaches, and in the case of full solution with SGA the solution quality is 4% worse than the solution

attained by e-SAHGA. Fig. 6.7 summarizes the computational savings for domain decomposition when using SGA or e-SAHGA. Overall the results demonstrate that the decomposition domain approach is substantially faster than the full approach and, most importantly, the solution quality is not affected for this problem. Note also that both approaches required fewer function evaluations than predicted (compare Tables 6.2 and 6.3 with the predictions in Table 6.1). This occurred because the predictions are based on the maximum number of generations (see Table 6.1)), but the runs converged prior to that generation.




**Injection Points:** **SGA: O** **e-SAHGA: □**

**Fig. 6.6 Full approach: Convergence ratio**

**Table 6.4 Summary of Results:**
**Number of Function Evaluations**

| | Algorithm | Solution Approach | | Difference |
|---|---|---|---|---|
| | | Domain Decomposition | Full | |
| **Number of Function Evaluations** | **SGA** | 2,910 | 12,000 | **75.8%** |
| | **e-SAHGA** | 1,968 | 5,169 | **61.8%** |
| **Solution Quality: Optimal Cost** | **SGA** | 1,911.8 | 1,986.7 | **4.00%** |
| | **e-SAHGA** | 1,911.2 | 1,911.1 | **0.01%** |

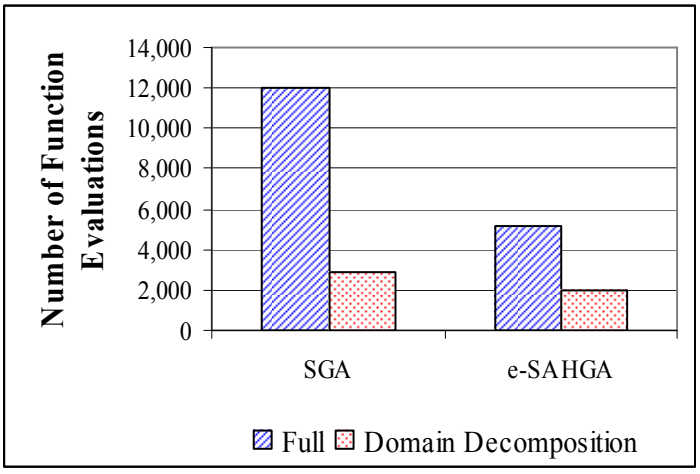



**Fig. 6.7 Number of function evaluations for domain decomposition and full approach**

## 6.4 Conclusions

The results presented in this chapter show that the e-SAHGA algorithm is capable of solving the complex problem of field-scale groundwater remediation design in an effective way. A domain decomposition approach was evaluated, in which well locations are identified first and then pumping rates in separate GA runs. The domain decomposition approach was shown to be much faster than the full solution approach with no loss in accuracy of the final solution; savings were 75.8% for SGA and 61.9% for e-SAHGA. For both solution approaches, e-SAHGA was substantially faster than the traditional SGA, with savings of 32.4% for the decomposed approach and 59.6% for the full approach. Moreover, the SGA failed to identify the best solution that e-SAHGA found in the full approach for a much larger population size. Further testing of the domain decomposition approach is needed on other remediation design problems to identify whether similar savings are obtained without loss of accuracy.

# Chapter 7

## CONCLUDING REMARKS

### 7.1 Summary of Research Findings

This research has developed a highly reliable numerical tool, the enhanced self-adaptive hybrid genetic algorithm (e-SAHGA) to more efficiently and effectively solve problems using genetic algorithms (GAs). With this tool, the designer can evaluate different solution alternatives in a more timely fashion. A step-by-step methodology has also been developed for evaluating the optimal parameters for using the algorithm. This methodology, together with the adaptive nature of the algorithm, reduces the need for trial-and-error experiments to determine the optimal set of parameters for the algorithm.

The reliability and reduced use of computer resources were first demonstrated by applying the original SAHGA algorithm to eight traditional problems from the Genetic and Evolutionary Algorithms field (see Chapter 3). For these problems, the algorithm is more than 90% reliable (in terms of identifying the correct optimal solution), with computational savings ranging from 7% to 47% with respect to the SGA in 100 runs with different random initial populations for most of the local search algorithms. Then, in Chapter 5, the enhanced algorithm (e-SAHGA) applied to a groundwater remediation design problem showed 90% reliability in identifying the solution faster than the SGA, with average savings of 64% across 100 runs with different random initial populations. Finally, e-SAHGA was tested on a field-scale remediation design problem, re-evaluation of the remediation system for Umatilla Army Depot (Chapter 6),

where it gave computational savings between 30% and 60% and, for one solution method, found a solution that was 4% better than the one found by the SGA. Due to the computational effort associated with the Umatilla case, only one initial random population was used; results may vary for other initial populations.

Another finding of this research is the importance of the local search algorithm selected for use within the hybrid algorithm. The results show that the same local search algorithm does not always give the best performance across all test functions and the groundwater remediation test case, primarily because of the difference in effort required to apply the different local search algorithms. For any function, the most suitable local search algorithm can be pre-selected using only the population sizing analysis shown in Section 3.4. When the difference in population sizes for different local search algorithms is not significant, which occurs when the algorithms give similar reductions in population standard deviation, it is more likely that a random search algorithm (which requires only one function evaluation per local search iteration) will perform better than steepest descent (which requires $n$+1 function evaluations per iteration, where $n$ is the number of decision variables) or random derivative (which requires 2 function evaluations per iteration).

Another important result is related to the population sizing approach presented in Section 3.4. A sequential injection approach was proposed in Section 4.3.3 for the groundwater remediation problem, where the smallest population size is run first, the optimal solution is injected into the initial population at the next population size, and the process continues until subsequent population sizes obtain the same solution. This approach resulted in 45% savings in

computational effort for the groundwater remediation test case using the original SAHGA algorithm. This sequential process ensures that a sufficient but not excessively large population size is used, achieving optimal performance. With respect to the HGA, the analysis clearly shows how local search reduces the standard deviation of fitness in the population, which in turn reduces the required population size to achieve the same level of reliability.

To improve performance of SAHGA further, a number of enhancements to the original algorithm were proposed and tested in Chapter 5. The most effective modifications were a fitness-based clustering approach to select the individuals undergoing local search and a new mechanism to stop local search when it is no longer needed. Combining these enhancements improved average computational effort associated with 100 runs of the groundwater remediation test case from 14% to 64% savings over the SGA, as well as improving reliability of the algorithm in performing faster than the SGA from 80% to 90%. The success of these enhancements highlights the importance of using local search only where it is most needed, since the additional function evaluations required for local search can increase computational effort over the SGA if they are not done effectively. The enhancements introduced to the original algorithm in Section 5.2 to create the e-SAHGA do not work with problem-specific information; hence, the enhanced algorithm can be applied to other types of optimization problems.

Finally, Chapter 6 evaluated a domain decomposition approach on the Umatilla field-scale remediation design problem. In this approach, well locations are identified first and then pumping rates are identified subsequently in separate GA runs. The domain decomposition approach was shown to be much faster than the full solution approach with no loss in accuracy of

the final solution for this problem; savings were 75.8% for SGA and 61.9% for e-SAHGA. For both approaches, e-SAHGA was substantially faster than the traditional SGA, with savings of 32.4% for the decomposed approach and 59.6% for the full approach. Moreover, the SGA failed to identify the best solution that e-SAHGA found in the full approach for a much larger population size.

## 7.2 Future Research

The first natural extension of this research would be the inclusion of local search in well locations by means of neighborhood search, in addition to the local search in pumping rates investigated in this study. In this local search scheme, well locations would be exchanged among the current location and one of the neighbor wells. The search could be performed sequentially, searching the neighborhood following a predefined sequence, or by randomly selecting the new location. Then, a combined search could be performed with local search in both locations and/or pumping rates.

A second extension to the algorithm would be the combination of different local search schemes (for pumping rates) in order to maximally exploit the potential of each algorithm. For example, at the beginning of the search, a random search algorithm is better for looking for promising regions in the decision space. Later in the search, a gradient-based algorithm (a more computationally expensive algorithm) may be better for quickly honing in on the solution because the promising regions in the decision space are already found.

Another possibility for extension of this research would be to include concepts from the memetic algorithms field in the hybrid GA. In memetic algorithms, local search is embedded in the GA processes and the result from the local search process is shared among the individuals in the population. This process guarantees that all of the individuals learn about good and bad regions in the space, which should speed up the search for the optimal solution.

A fourth potential line of research would be to modify the algorithm to consider uncertainty by means of a noisy HGA. In this case, the true value of one or more of the parameters used to define the problem is unknown, and we only know the possible range of variation of the parameter. For example, for a groundwater remediation problem we can evaluate the hydraulic conductivity at very specific points of the system. Using these point values, it is possible to evaluate different spatial distributions for this parameter. Now, the noisy HGA would by evaluating the candidate solutions not only for one possible distribution of the hydraulic distribution but for many of them. In this way, it is possible to evaluate how good the candidate solution is for different conditions. Smalley et al. (2000) and Gopalakrishnan et al. (2003) investigated the noisy GA for groundwater remediation, demonstrating that such an approach can identify highly reliable solutions with few samples of each solution. In the case of the noisy HGA, however, sampling should be included not only in the GA step but also at the local search step. Including sampling may affect the performance and operations of the HGA algorithm, particularly the efficacy of different local search algorithms.

The final natural application of the e-SAHGA is the creation of a multiobjective e-SAHGA. In this case, the algorithm would be capable of evaluating conflicting objectives

simultaneously. For example, for a groundwater remediation problem, some of the objectives to study could be cost, risk, cleanup time, or mass removal. The conflicting objectives are linked by means of the so-called Pareto front that defines how one objective is related with other; along the Pareto front, no solution can be improved in one objective without sacrificing performance on another objective. Traditionally, the evaluation of this Pareto front requires large populations. For this reason, the application of local search could help to speed up the process of evaluating the front by using a smaller population size to achieve the same (or similar) result when we compare with the multiobjective GA without local search.

## References


Aguado, E., Remson, I. Pikul, M. F., and Thomas, W. A. (1974). "Optimum pumping for aquifer dewatering." *Journal Hydraulics Division,* ASCE, 100, 860-877.

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

Alabau, M., Idoumghar, L., and Schott, R. (2002). "New hybrid genetic algorithms for the frequency assignment problem." *IEEE Transactions on Broadcasting,* 48(1), 27-34.

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.

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

Bäck, T., D. Fogel and Z. Michalewicz, (eds): Handbook of Evolutionary Computation, Bristol and New York. Institute of Physics Publishing Ltd and Oxford University Press (1997).

Baldwin, J. M. (1896). "A new factor in evolution." *American Naturalist*, 30, 441-451.

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

Bracken, J. G. P. McCormick. (1970). *Ausgewählte Nwendungen Nichtlinearer Programmierung*. Berliner Union and Kohlhammer, Sttutgart.

Branin, F. K. (1972). "A widely convergent method for finding multiple solutions of simultaneous nonlinear equations." *IBM J. Res. Develop*., 504-522.

Cai, X., McKinney, D. and Lasdon. L. (2001). "Solving nonlinear water management models using a combined genetic algorithm and linear programming approach." *Advances in Water Resources,* 24, 667-676.

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.

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.

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

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

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

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

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

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

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

Dawkins, R, (1982). *The extended phenotype: The gene as the unit of selection*. Oxford, UK: W H Freeman.

Dougherty, D. E., and Marryott, R. A. (1991). "Optimal groundwater management. 1. Simulated annealing." *Water Resources Research,* 27(10), 2493-2508.

De Jong, K. A., (1975). "An analysis of the behavior of a class of genetic adaptive systems." Ph.D. dissertation, University of Michigan, Ann Arbor, MI.

Dixon, L. C. W. and Szego, G. P. (1978). The Optimization Problem: An Introduction. In Dixon, L. C. W. and Szego, G. P. (Eds.): Towards Global Optimization II, New York: North Holland.

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

Environmental Security Technology Certification Program (ESTCP). (2003). Draft First Technical Report: Application of Flow and Transport Optimization Codes to Groundwater Pump and Treat Systems.

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.

Espinoza, F., B. S. Minsker, and D. Goldberg. (2001). "A self adaptive hybrid genetic algorithm", in *L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'2001*. San Francisco, Morgan Kaufmann Publishers.

Eusuff, M. and Lansey, K. (2003). "Optimization of water distribution network design using the shuffled frog leaping algorithm." *Journal of Water Resources Planning and Management,* 129(3), 210-225.

Foster, N., and Dulikravich, G. (1997). "Three-dimensional aerodynamic shape optimization using genetic and gradient search algorithms", *Journal of Spacecraft and Rockets,* 34(1), 36-42.

Gailey, R. (1999). "Application of mixed-integer linear programming techniques for water supply management and plume containment at a California EPA site." *Proc. 26$^{th}$ Annu. Water Resour. Plng. And Mgmt. Conf.,* ASCE, Reston, VA.

Gen, M., Ida, K., and Li, Y. (1998). "Bicriteria transportation problem by hybrid genetic algorithm." *Computers & Industrial Engineering*, 35(1-2), 363-366.

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

Goldberg, D. E. and Voessner, S. (1999). "Optimizing global-local search hybrids." *ILLIGAL report 99001.*

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

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

Harbaugh, A.W., Banta, E.R., Hill, M.C., and McDonald, M.G. (2000) *MODFLOW-2000, the U.S. Geological Survey modular ground-water model--user guide to modularization concepts and the ground-water flow process: U.S. Geological Survey Open-File Report 00-92.*

Harik, G.R., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L. (1997). "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.

Hart, K. A., (1994). "Adaptive global optimization with local search." Ph.D. dissertation, University of California, San Diego.

Hinton, G. E., and Nolan, S. J. (1987). "How learning can guide evolution." *Complex Systems*, 1:495-502.

Hogg, R., and Craig, A. (1978) *Introduction to mathematical statistics.* Macmillan Publishing Co., Inc., New York.

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

Hsiao, C. and Chang, L. (2002). "Dynamic optical groundwater management with inclusion of fixed costs." *Journal of Water Resources Planning and Management,* 128(1), 57-65.

Iman, R. L., Davenport, J. M., and Zeigler, D. K. "Latin hypercube sampling (a programs user guide)." (1980). Technical report SAND79-1473, Sandia Laboratories, Alburquerque.

Kapelan, Z., Savic, D., and Walters, A. (2002). "Hybrid GA for calibration of water distribution hydraulic models." *2002 Conference on Water Resources Planning and Management*, Roanoke Va

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

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

Kazarlis, S., Papadakis, S., Theocraris, J., and Petridis, V. (2001). "Microgenetic algorithms as generalized hill-climbing operators for GA optimization", *IEEE Transactions in Evolutionary Computation,* 5(3), 204-217.

Kim, J-H. and H. Myung. (1997). "Evolutionary programming techniques for constrained optimization problems." *Evolutionary Computation*, 1(2), 129-140.

Krishnakumar, K. (1989) "Micro-Genetic algorithms for stationary and non-stationary function optimization," SPIE: Intelligent Control and Adaptive Systems, Vol. 1196, Philadelphia, PA.

Lamarck, J. (1802). *Zoological philosophy: an exposition with regard to the natural history of animals.* University of Chicago Press.

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

Lin, W., Delgado-Frias, J, Gause, D., and Vassiliadis, S. (1995). "Hybrid Newton-Raphson genetic algorithm for the traveling salesman problem." *Cybernetics & Systems,* 26(4), 387-412.

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

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

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

Lobo, F. and Goldberg D. E.(1997). "Decision making in a hybrid genetic algorithm." In *Proceedings of the 1997 IEEE Conference on Evolutionary Computation",* 121-125.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observation. pp. 281-297 in: L.M. Le Cam & J. Neyman (eds.) *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. University of California Press, Berkeley.

Magyar, G., Johnsson, M., and Nevalainen, O., (2000). "An adaptive hybrid genetic algorithm for the three-matching problem." *IEEE Transactions on Evolutionary Computation,* 4(2), 135-146.

Mansfield, C. M. Shoemaker, C. A., and Liao, L-Z. (1998). "Utilizing sparsity in time-varying optimal control of aquifer cleanup". *Journal of Water Resources Planning and Management,* 124(1), 39-46.

Marryott, R. A., Dougherty, D. E., and Stollar, R. L. (1993). "Optimal groundwater management 2. Application of simulated annealing to a field-scale contamination site." *Water Resources Research,* 29(4), 847-60.

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.

Mathias, K.E.; Whitley, L.D.; Stork, C.; Kusuma, T. **"**Staged hybrid genetic search for seismic data imaging", *IEEE World Congress on Computational Intelligence. Proceedings of the First IEEE Conference on Evolutionary Computation*, 1994. 27-29.

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

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

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

Merz, P., and Freisleben, B. (2000). "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem." *IEEE Transactions on Evolutionary Computation,* 4(4), 337-352.

Minsker, B. S., and Shoemaker, C. A. (1998a). "Computational issues for optimal in-situ bioremediation design." *Journal of Water Resources Planning and Management,* 124(1), 39-46.

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

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

Moscato, P. (1989). "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms." *Caltech Concurrent Computation Program,* California Institute of Technology, Pasadena, Tech. Report 790.

Moscato, P., and Cotta, C. (1999). A gentle introduction to memetic algorithms, in handbook of metaheuristics, ed. Glower F. and Kochenberger G. 1-56, Kluwer.

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

Nguyen, H., Yoshihara, I., Yamamori, K., Yasunaga, M**. "**A parallel hybrid genetic algorithm for multiple protein sequence alignment" *Proceedings of the 2002 Congress on Evolutionary Computation*. 1, 12-17.

Park Y., Park J., and Won, J. (1998). "A hybrid genetic algorithm/dynamic programming approach to optimal long-term generation expansion planning." *International Journal of Electrical Power & Energy Systems*, 20(4), 295-303.

"RACER 99: Remedial Action Cost Engineering and Requirements Users Guide." (1999) Talisman Partners, Ltd.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologishen evolution.* Frommann-Iolzboog Verlag, Stuttgart.

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

Reed, P. (2002). "Striking the balance: long-term groundwater monitoring design for multiple conflicting objectives." PhD dissertation, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign.

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

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

Sabatini, A. (2000). "A hybrid genetic algorithm for estimating the optimal time scale of linear systems approximations using Laguerre models." *IEEE Transactions on Automatic Control,* 45(5), 1007-1011.

Sawyer, C. S., and Ahlfeld, D. P. (1992). "A mixed-integer model for minimum cost of remediating a multilayer aquifer". *Computational methods in water resources: Numerical methods in water resources,* T. F. Russell et al., eds., Elsevier Applied Science, London, England, 353-360.

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

Schönberger, J. Mattfeld, D. C., and Kopfer, H. (in press). "Memetic algorithm timetabling for non-commercial sport leagues." *European Journal of Operational Research.*

Schwefel, H. P. (1981). *Numerical optimization of computer models*. John Wiley & Sons, Chichester-New York-Brisbane-Toronto.

Schwefel, H. P. (1975). “Evolutionsstrategie und numerische Optimierung“. PhD dissertation, Department of Process Engineering, Technical University of Berlin, Berlin, Germany.

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.

Sinclair, M. (1999). “Minimum cost wavelength-path routing and wavelength allocation using a genetic-algorithm/heuristic hybrid approach.” *IEEE Proceedings Communications,* 146(1).

Sinha, A., and Goldberg, D. E. (2001). Verification and extension of the theory of global-local hybrids, *in* L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'2001. San Francisco, Morgan Kaufmann Publishers.

Srivastava R. P. (2002). “Time Continuation in Genetic Algorithms.” Master's thesis from the University of Illinois.

Srivastava, R.P., Goldberg, D.E. (2001). Verification of the Theory of Genetic and Evolutionary Continuation, *in* L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'2001. San Francisco, Morgan Kaufmann Publishers.

Spitzer, F. (1964). *Principles of random walk.* D. Van Nostrand Company, Inc.

Taguchi, T., Yokota, T., and Gen, M. (1998). "Reliability optimal design problem with interval coefficients using hybrid genetic algorithms." *Computers & Industrial Engineering*, 35(1-2), 373-376.

Tang. J., Wang, D., Ip, A., and Fung, R. (1998). "A hybrid genetic algorithm for a type of nonlinear programming problem." *Computers and Mathematics with Applications,* 36(5), 11-21.

Thierens, D., and Goldberg, D. E. (1994). "Convergence models of genetic algorithm selection schemes", In Y. Davidor, Hans-Paul Schwefel, and Reinhard Manner (eds) *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pp. 119-129, Springer-Verlag, New York, NY.

Thierens, D., Goldberg, D. E., and Guimaraes Pereira, A. (1998). "Domino convergence, drift, and the temporal-salience structure of problems", In *The 1998 IEEE International Conference on Evolutionary Computation Proceedings*, pp. 535-540, IEEE Press, New York, NY, 1998.

Thierens, D. (1995). "Analysis and design of genetic algorithms." PhD dissertation, Katholieke Universiteit Leuven, Leuven, Belgium.

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

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

Yu, D. M., Rizzo, M., and Dougherty, D. E. (1998). "Multi-period objectives and groundwater remediation using samoa: tandem simulated annealing and external cutting plane method for containment with cleanup." *XII International Conference on Computational Methods in Water Resources*, 332-340.

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.

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

Whitley, D., Gordon, V. S., and Mathias, K. (1994). "Lamarckian evolution, the Baldwin effect and function optimization." *Parallel Problem Solving from Nature- PPSN III,* 6-15.

Zheng, C., and Wang, P. P. (1999a). "An integrated global and local optimization approach for remediation system design." *Water Resources Research,* 35(1), 137-148.

Zheng, C., and Wang, P. P. (1999b). MT3DMS: A modular three-dimensional multispecies transport model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater systems; documentation and user's guide, Contract Report SERDP-99-1, U.S. Army Engineer Research and Development Center, Vicksburg, MS.

# APPENDIX A

# DETAILED RESULTS FOR TEST FUNCTIONS FROM CHAPTER 3 (ORIGINAL SAHGA ALGORITHM)

# APPENDIX A

## TEST FUNCTIONS

This Appendix presents the results for the analysis presented in Section 3.5. The results are presented for the 7 test problems not included in that section. These functions are:

- DJ1
- DJ2
- Branin
- Six-Hump
- Schwefel
- Test08
- B&M

Tables A.1 through A.3 detail the standard deviation reductions for all of the test problems and local search algorithms, together with the population size for the application of the HGA algorithm.

**Table A.1 Population size for different values of K (SGA)**

| Building Block Order (K) | Test Function | | | | | | |
|---|---|---|---|---|---|---|---|
| | DJ1 | DJ2 | Branin | Six-Hump | Schwefel | Test04 | B&M |
| 1 | 20 | 12 | 20 | 10 | 35 | 16 | 24 |
| 2 | 40 | 24 | 40 | 20 | 70 | 32 | 48 |
| 3 | 80 | 48 | 80 | 40 | 140 | 64 | 96 |
| 4 | 160 | 96 | 160 | 80 | 280 | 128 | 192 |
| 5 | 320 | 192 | 320 | 160 | 560 | 256 | 384 |

**Table A.2 Standard Deviation Reduction for Test Problems (β)**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 0.90 | 0.90 | 0.90 | 0.90 | 0.90 |
| **DJ2** | 0.67 | 0.67 | 0.67 | 0.42 | 0.42 |
| **Branin** | 0.60 | 0.60 | 0.50 | 0.30 | 0.30 |
| **Six-Hump** | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 |
| **Schwefel** | 0.86 | 0.80 | 0.77 | 0.57 | 0.57 |
| **Test08** | 0.94 | 0.88 | 0.88 | 0.88 | 0.63 |
| **B&M** | 0.96 | 0.92 | 0.92 | 0.50 | 0.50 |

**Table A.3 HGA Population for Test Problems**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 288 | 288 | 288 | 288 | 288 |
| **DJ2** | 64 | 64 | 64 | 40 | 40 |
| **Branin** | 96 | 96 | 80 | 48 | 48 |
| **Six-Hump** | 96 | 96 | 96 | 96 | 96 |
| **Schwefel** | 240 | 224 | 216 | 160 | 160 |
| **Test08** | 120 | 112 | 112 | 112 | 80 |
| **B&M** | 92 | 88 | 88 | 48 | 48 |

The performance and parameter effect for each algorithm (NAHGA and SAHGA) was evaluated by performing 100 runs with different random seeds, creating different initial populations. The default set of parameters for the analysis was selected based on previous experience working with test functions (Espinoza et al., 2001):

- Local search frequency: $\Delta G=3$ (NAHGA) and $T=0.75$ (SAHGA)
- Probability of local search: $P=0.1$ (NAHGA) and $P_0=0.1$ (SAHGA)
- Maximum number of local search iterations: $LS= 5$
- Adaptive parameter for local search probability (SAHGA): $\varepsilon= 0.1$
- Proportion of Baldwinian evolution: $PB= 0.25$

For every function, the results are presented in the following order:

- Adaptive parameter
- Probability of local search
- Local search effect: Frequency and Threshold
- Maximum number of local search iterations
- Proportion of Baldwinian evolution

# Test Function: DJ1

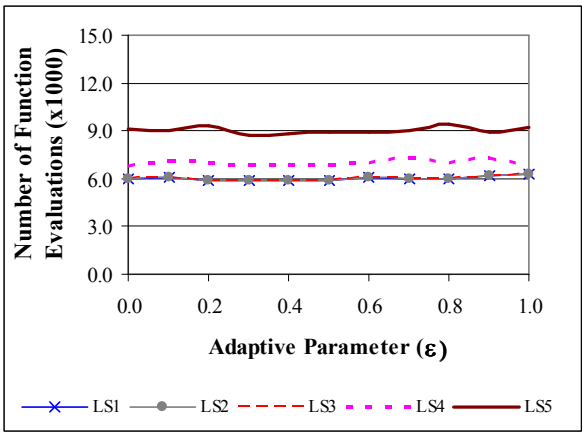



**Fig. A.1 Effects of adaptive parameter**

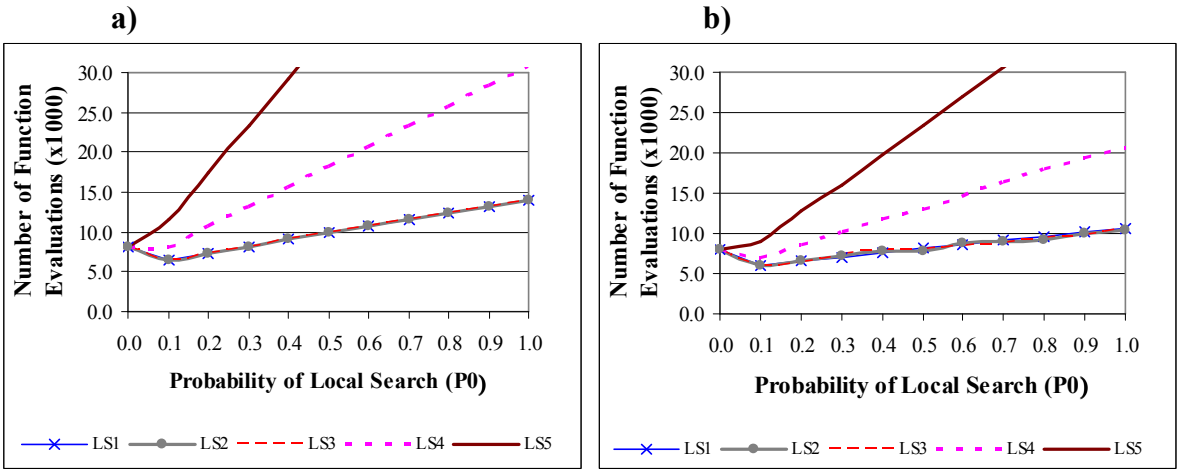



**Fig. A.2 Probability of local search: a) NAHGA, b) SAHGA**

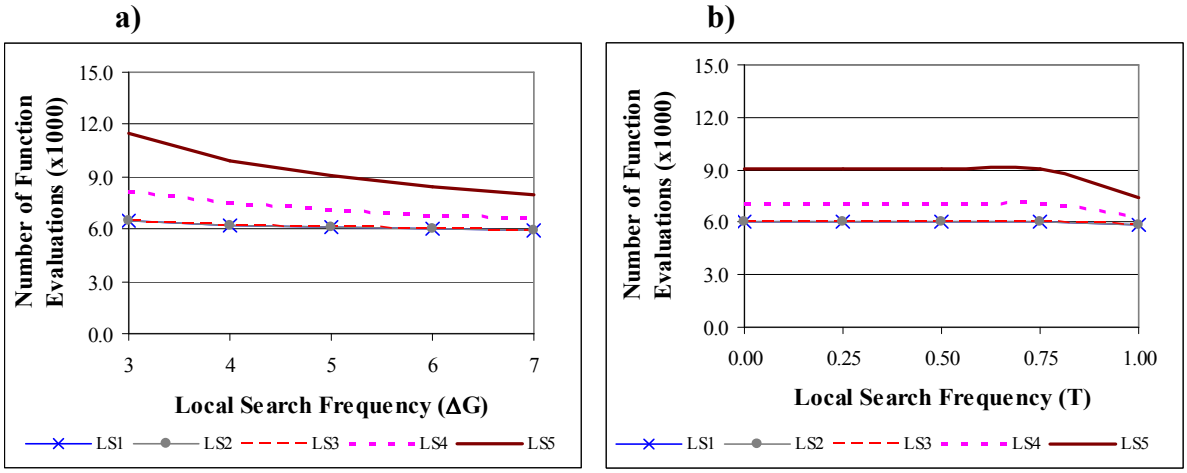



**Fig. A.3 Local search effect: a) NAHGA, b) SAHGA**

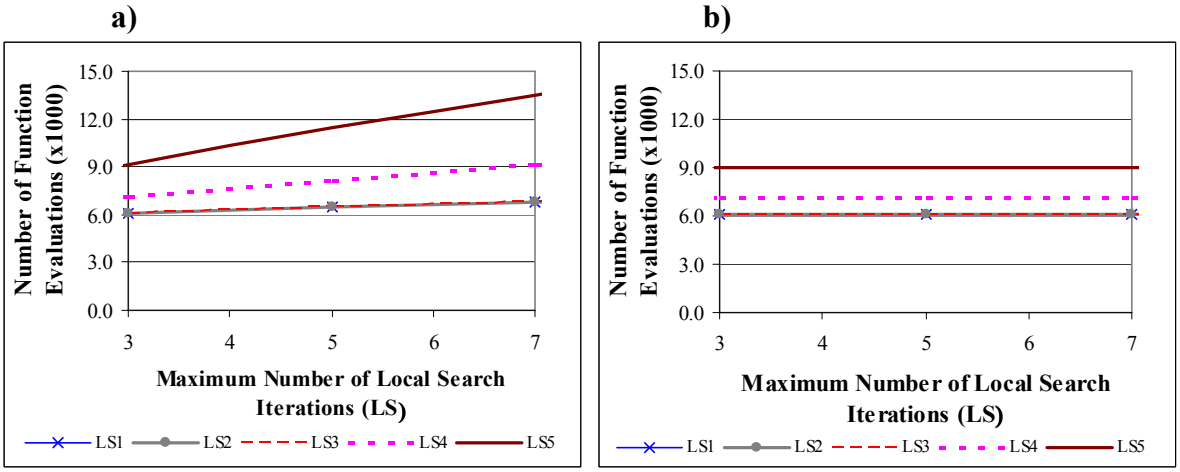



**Fig. A.4 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

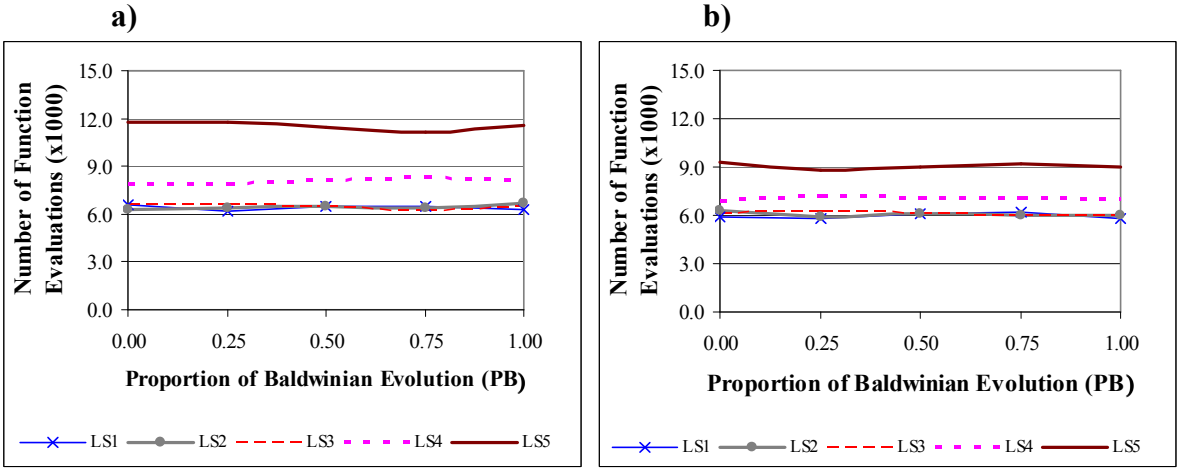



**Fig. A.5 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

# Test Function: DJ2

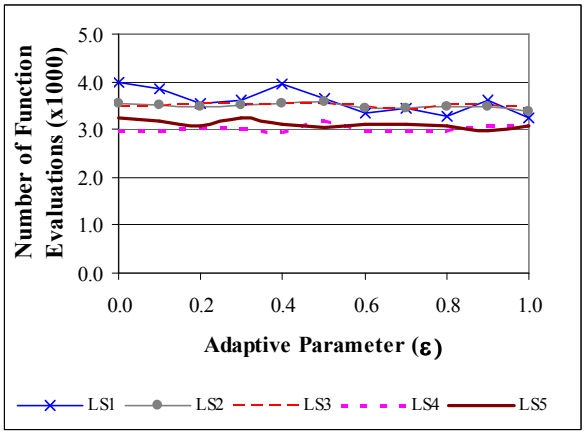



**Fig. A.6 Effects of adaptive parameter**

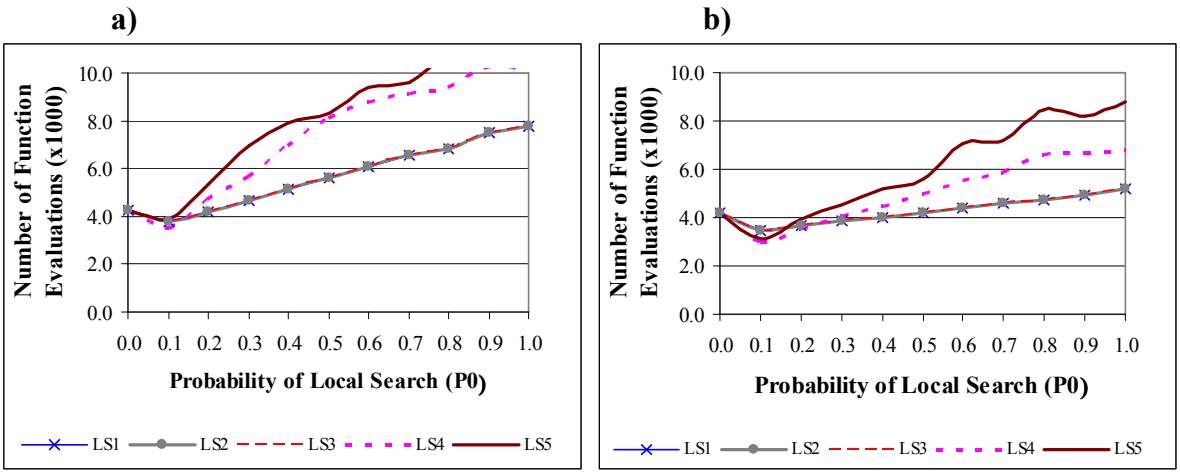



**Fig. A.7 Probability of local search: a) NAHGA, b) SAHGA**

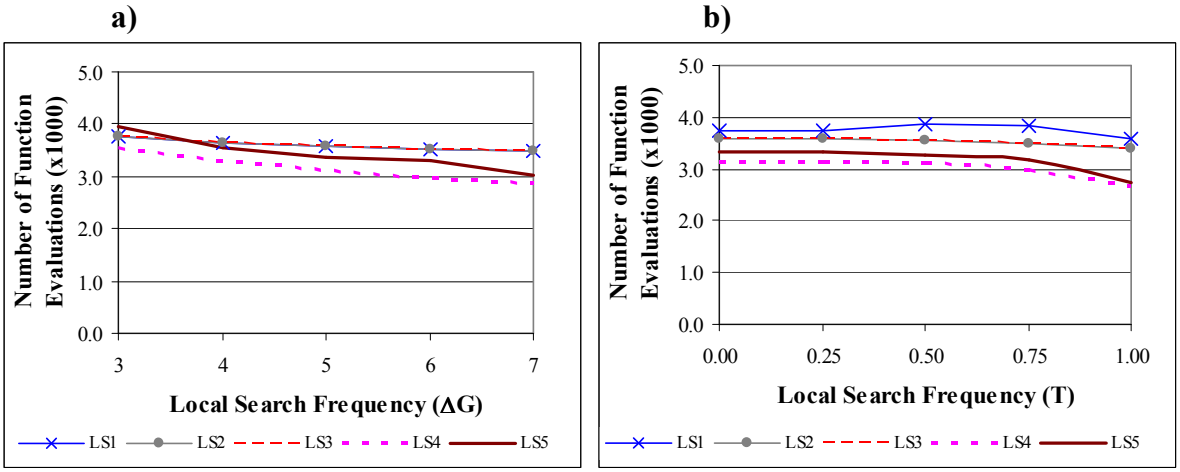



**Fig. A.8 Local search effect: a) NAHGA, b) SAHGA**

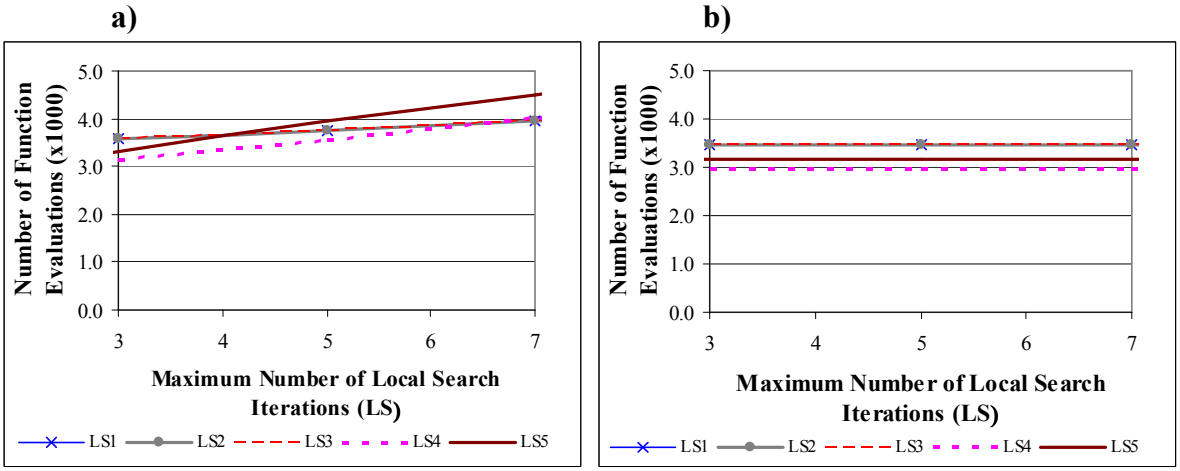



**Fig. A.9 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

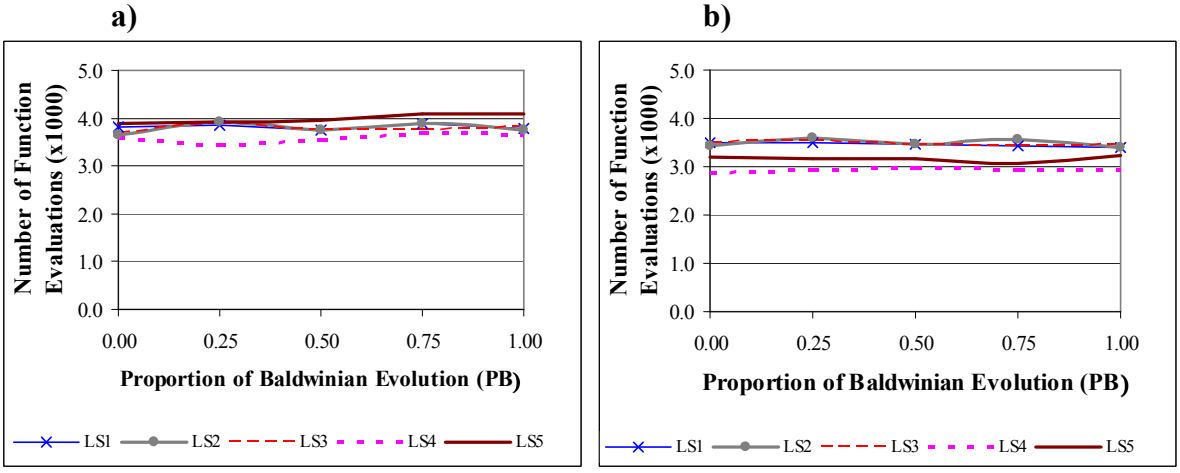



**Fig. A.10 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

# Test Function: Branin

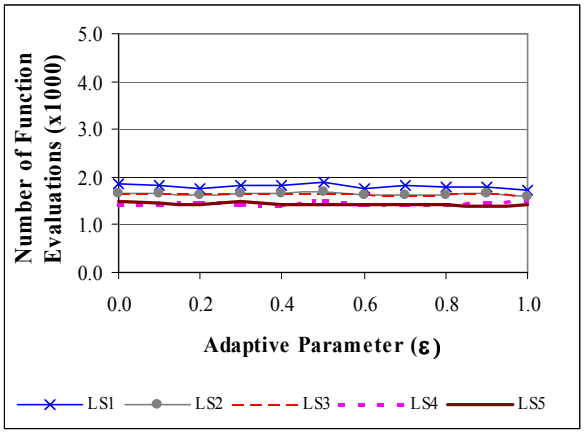



**Fig. A.11 Effects of adaptive parameter**

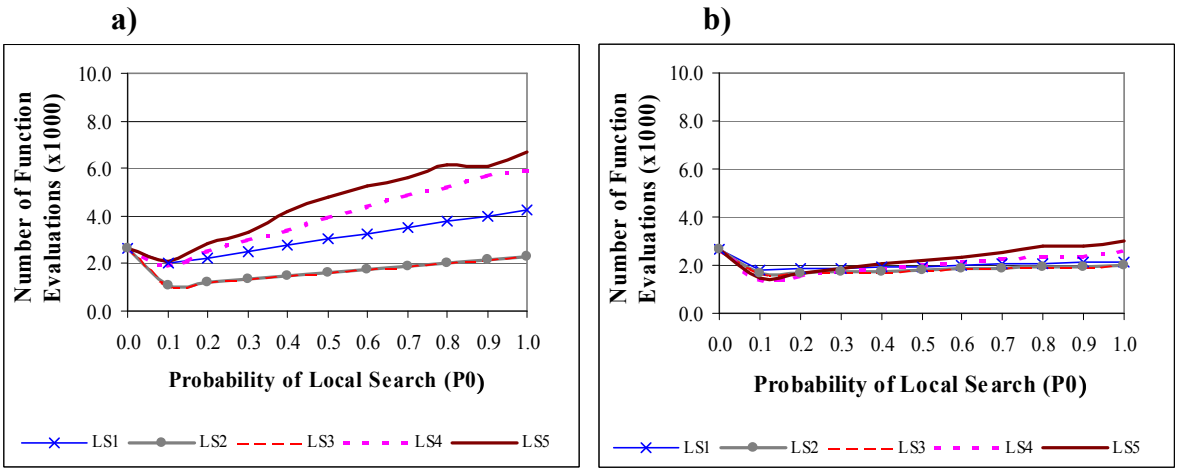



**Fig. A.12 Probability of local search: a) NAHGA, b) SAHGA**

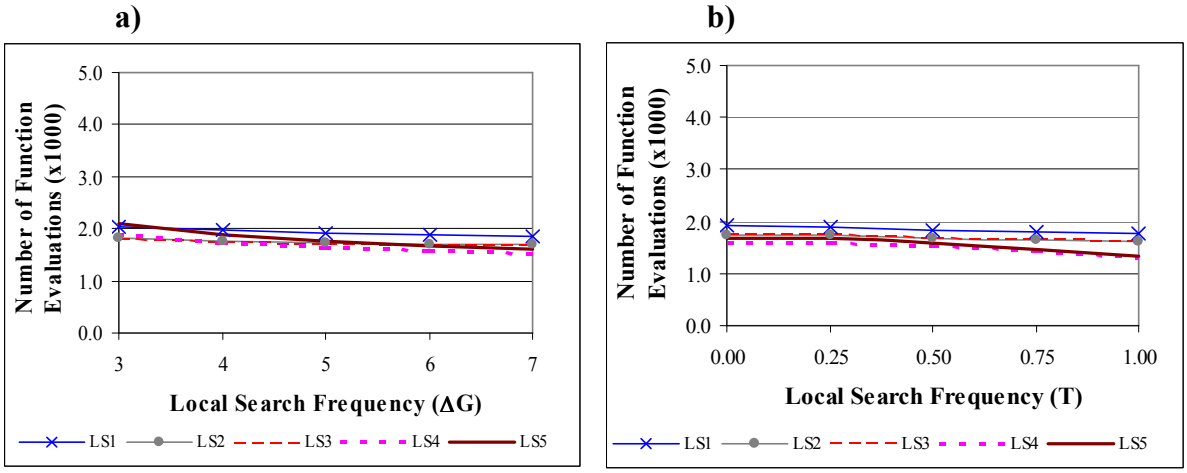



**Fig. A.13 Local search effect: a) NAHGA, b) SAHGA**

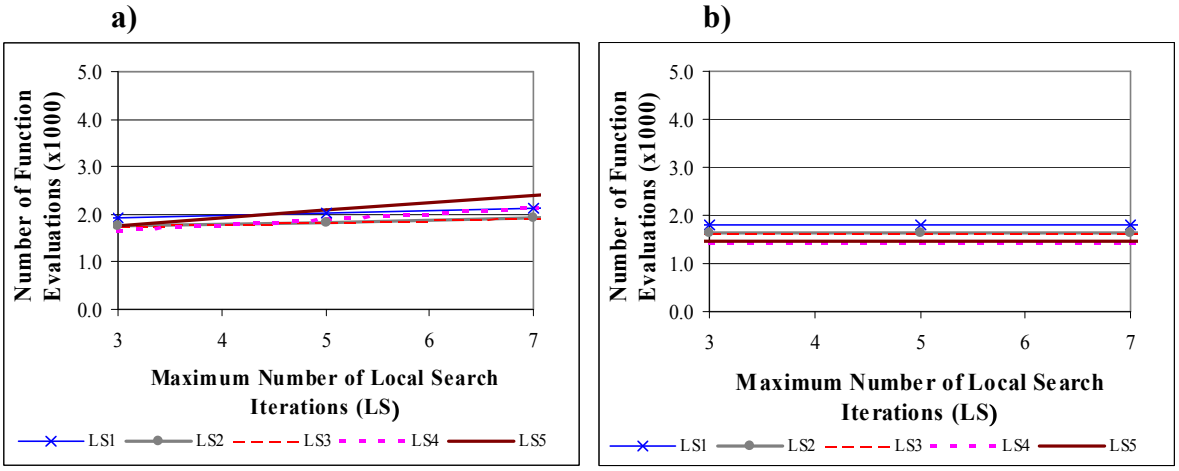



**Fig. A.14 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

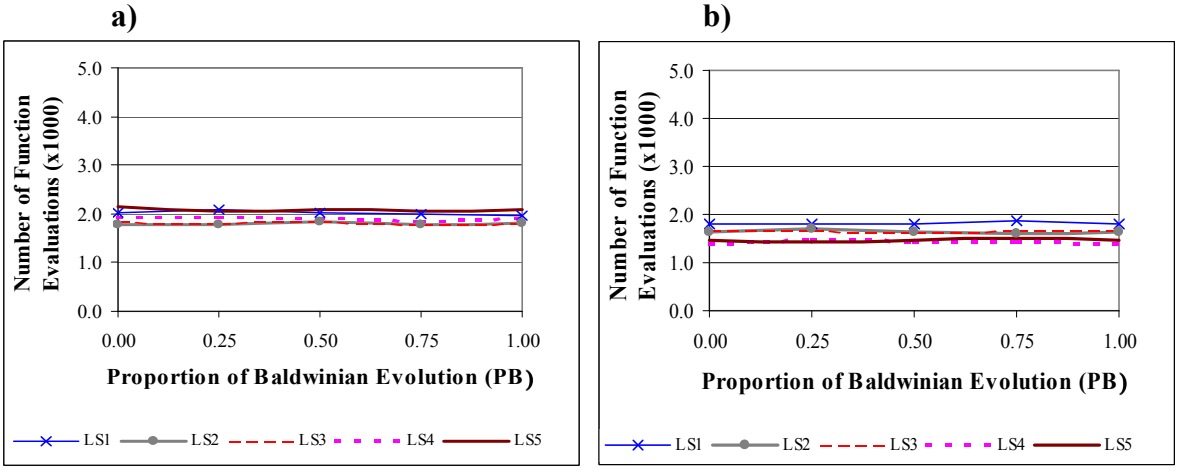



**Fig. A.15 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

# Test Function: Six-Hump

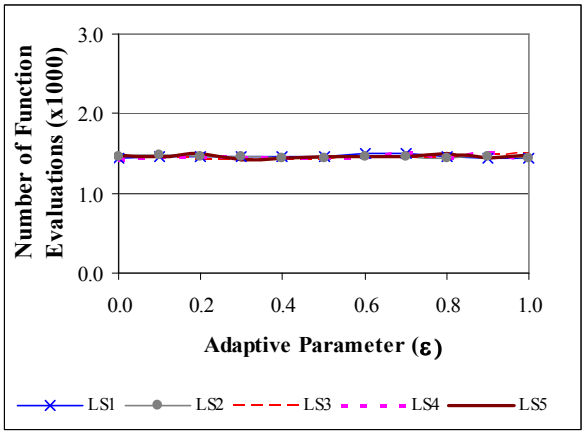



**Fig. A.16 Effects of adaptive parameter**

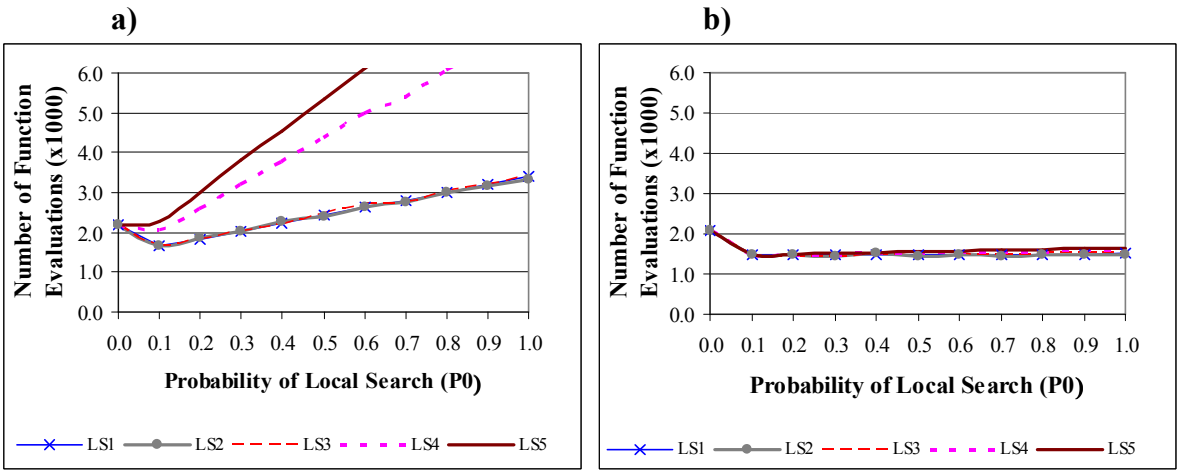



**Fig. A.17 Probability of local search: a) NAHGA, b) SAHGA**

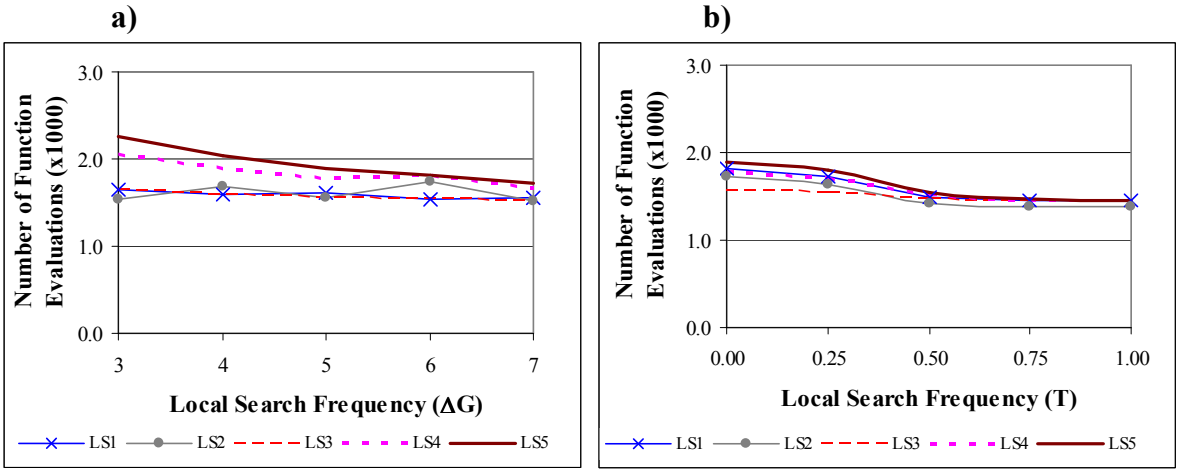



**Fig. A.18 Local search effect: a) NAHGA, b) SAHGA**

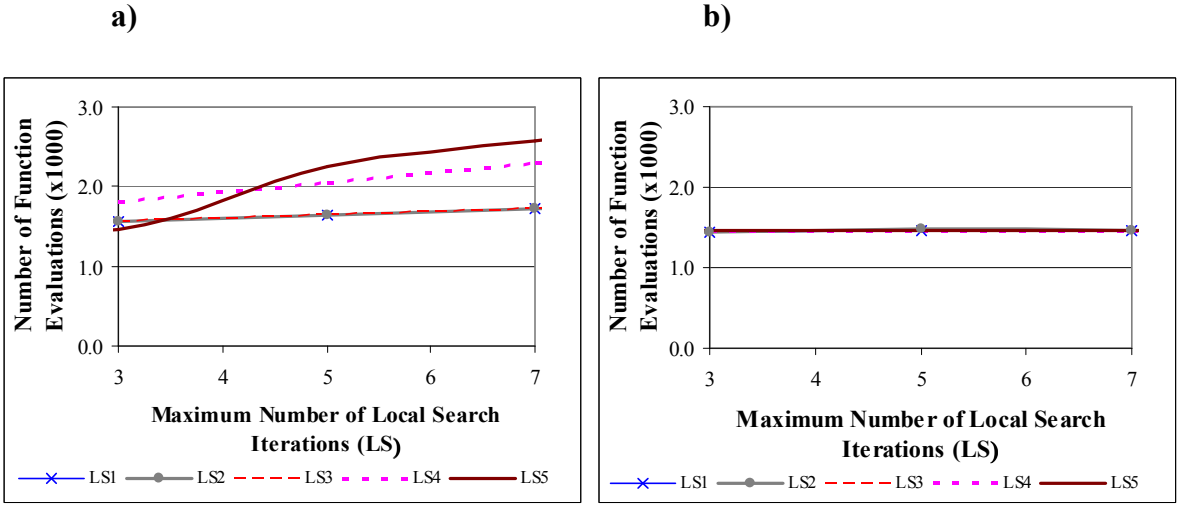



**Fig. A.19 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

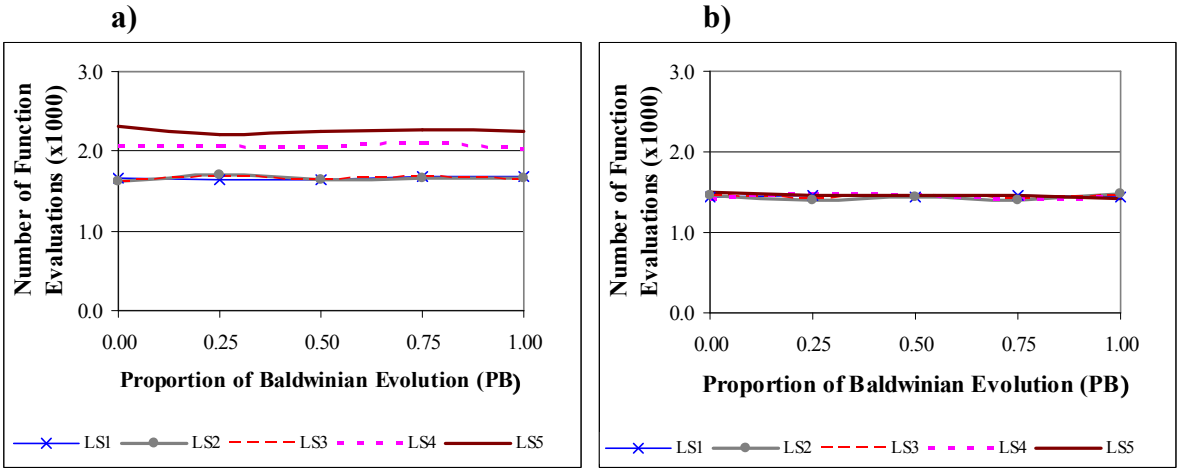



**Fig. A.20 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

# Test Function: Schwefel

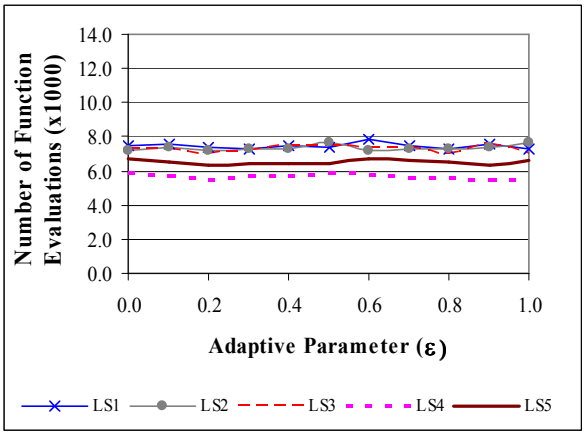



**Fig. A.21 Effects of adaptive parameter**

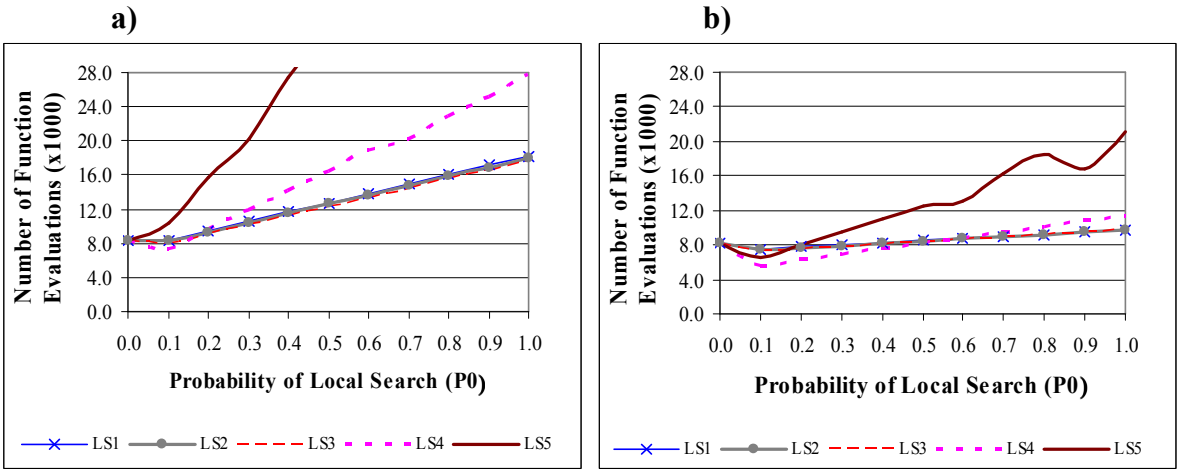



**Fig. A.22 Probability of local search: a) NAHGA, b) SAHGA**

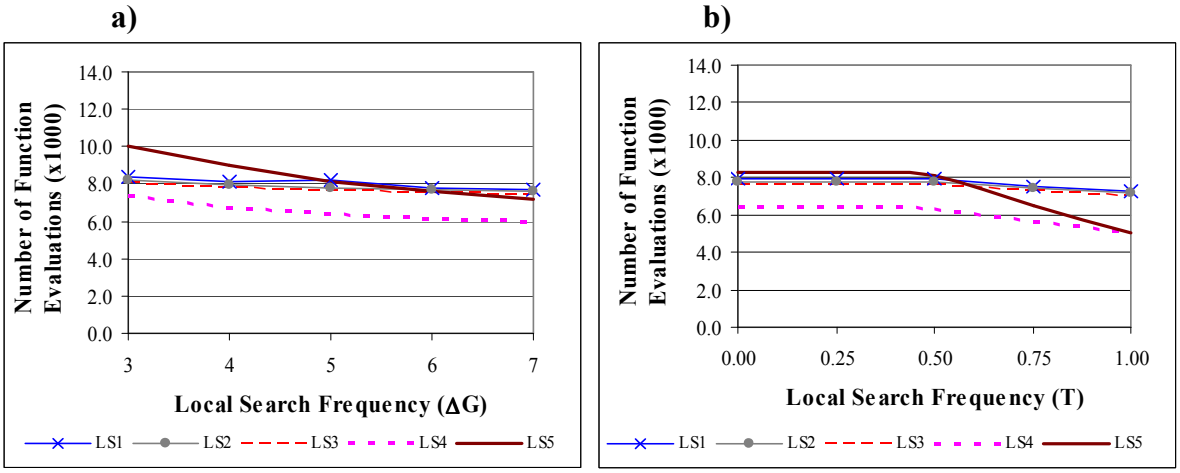



**Fig. A.23 Local search effect: a) NAHGA, b) SAHGA**

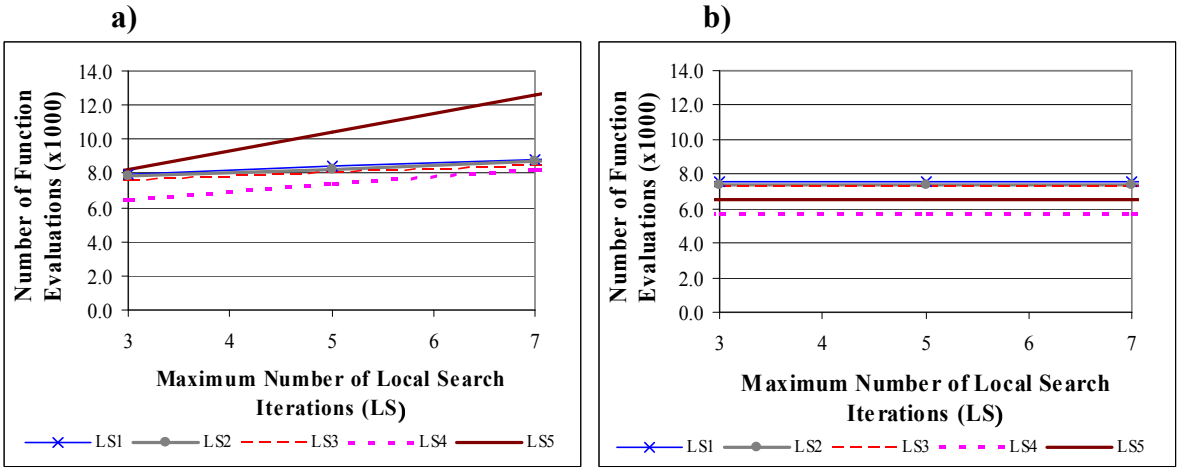



**Fig. A.24 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

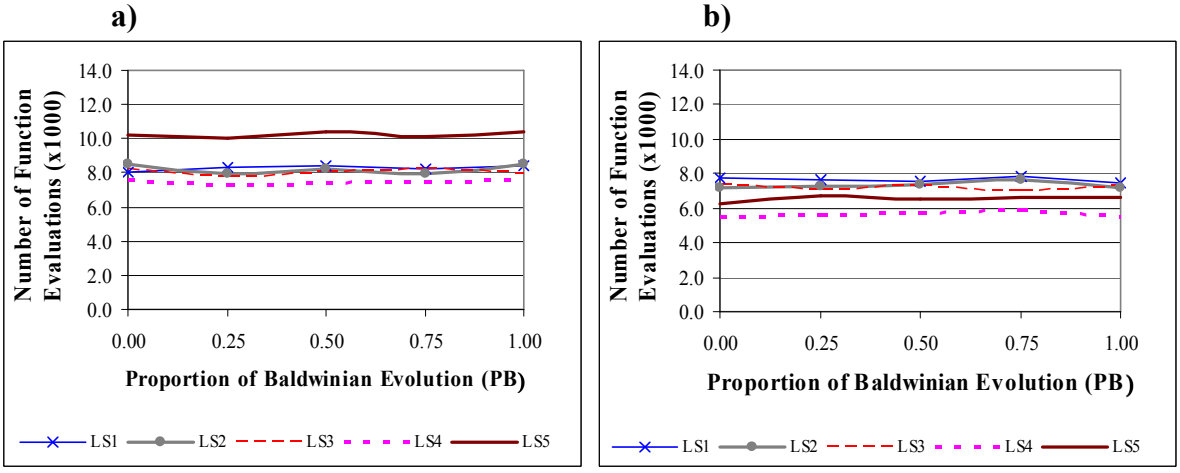



**Fig. A.25 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

# Test Function: Test08

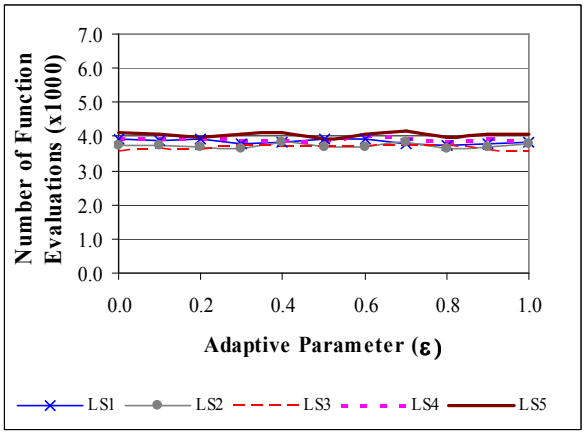



**Fig. A.26 Effects of adaptive parameter**

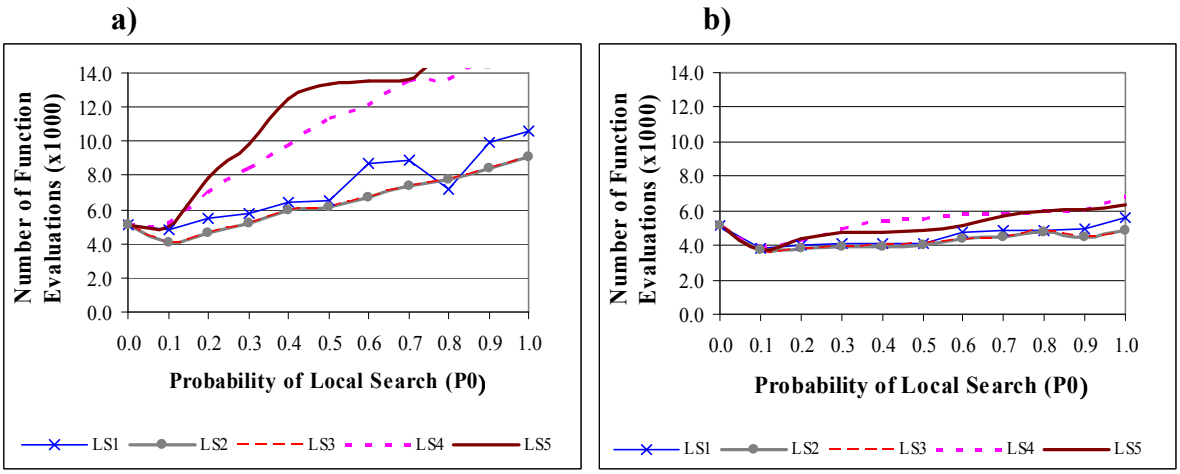



**Fig. A.27 Probability of local search: a) NAHGA, b) SAHGA**

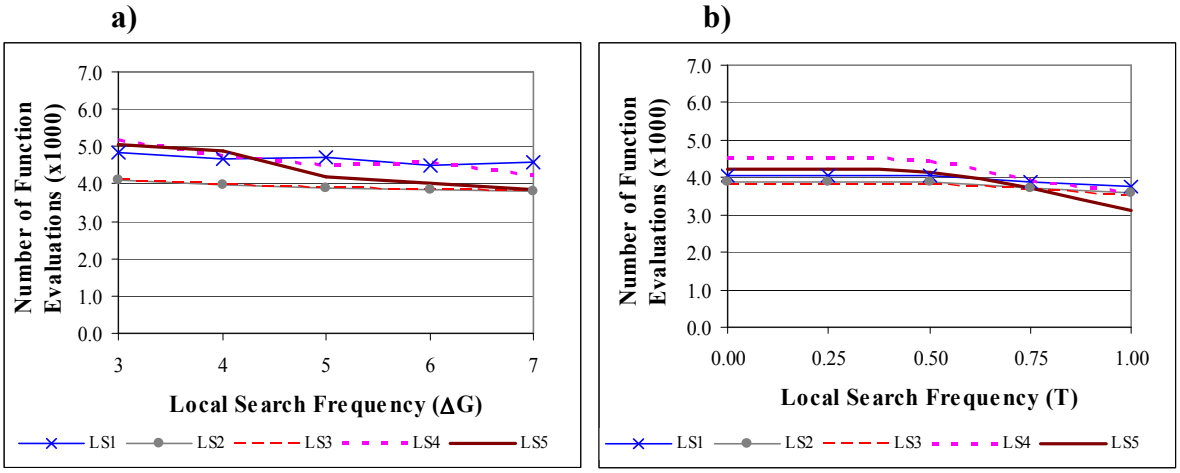



**Fig. A.28 Local search effect: a) NAHGA, b) SAHGA**

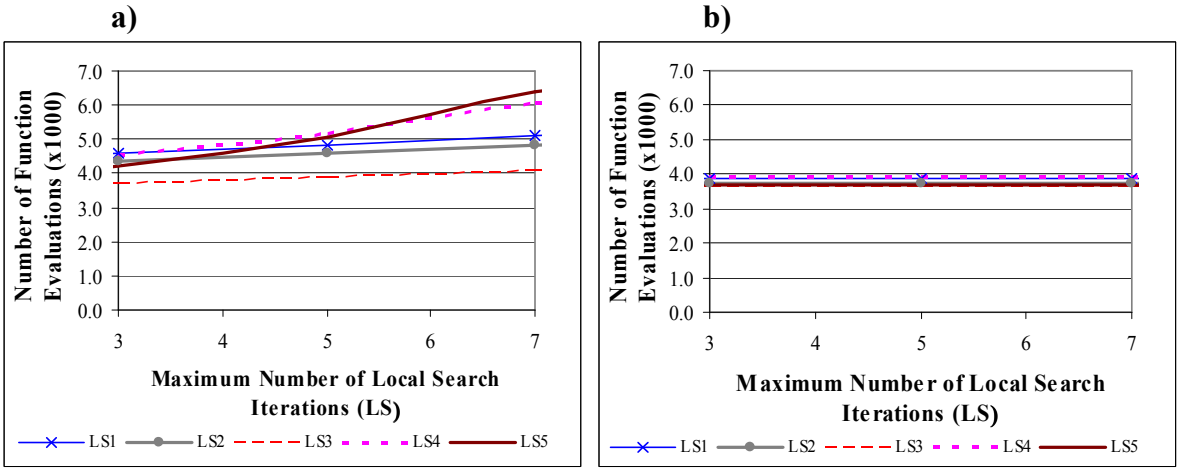



**Fig. A.29 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

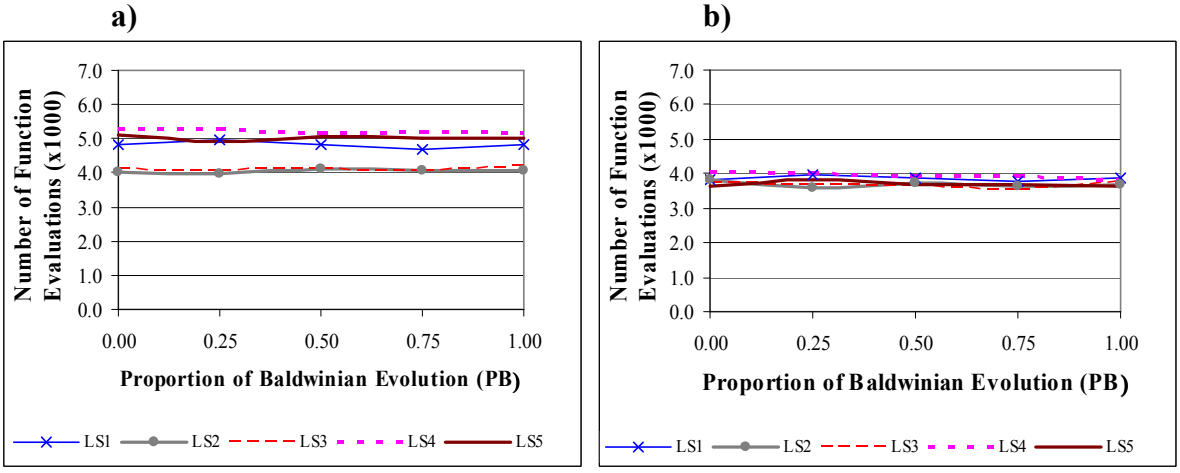



**Fig. A.30 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

# Test Function: B&M

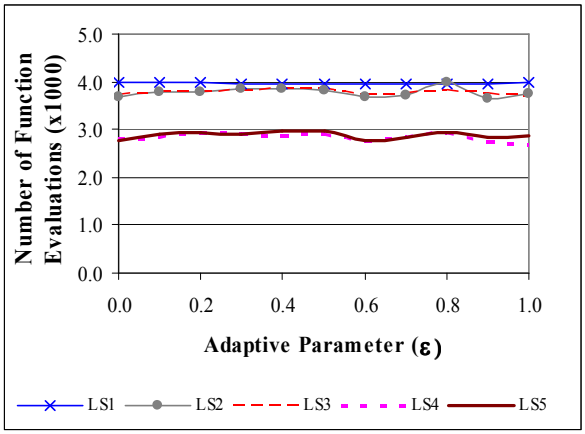



**Fig. A.31 Effects of adaptive parameter**

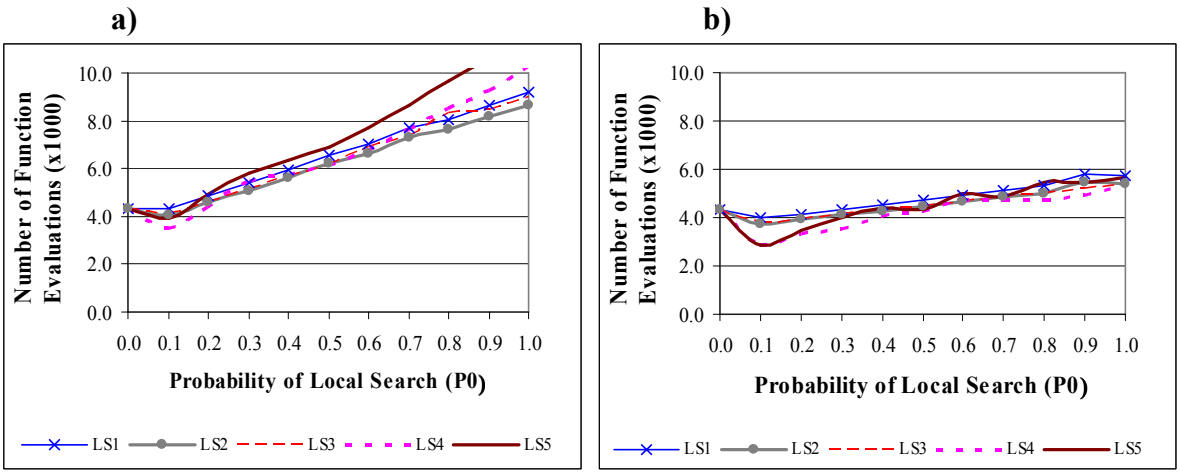



**Fig. A.32 Probability of local search: a) NAHGA, b) SAHGA**

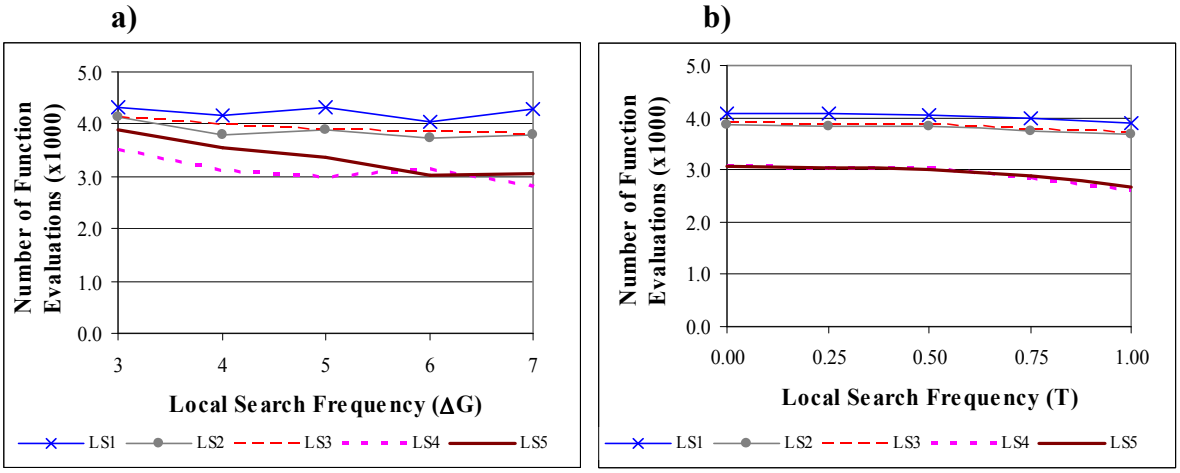



**Fig. A.33 Local search effect: a) NAHGA, b) SAHGA**

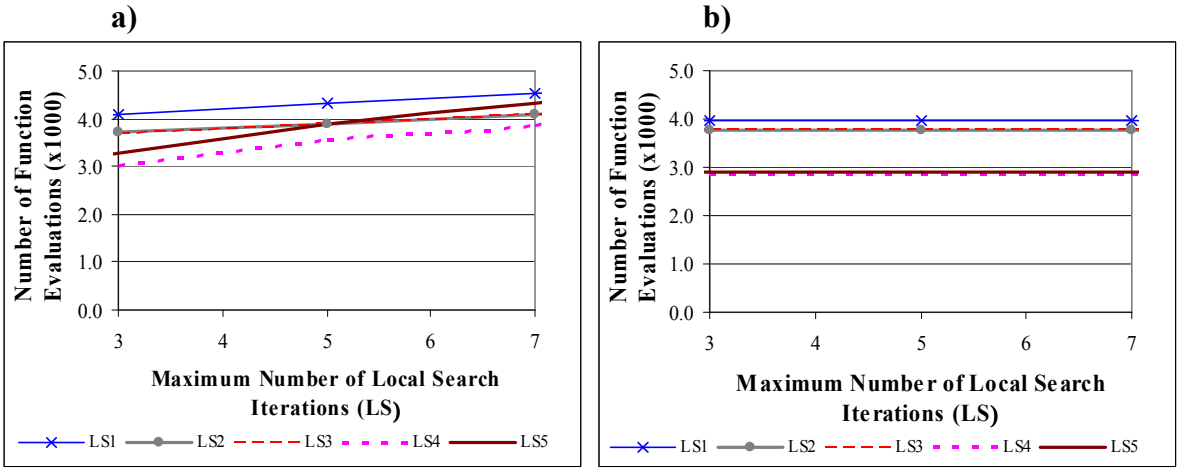



**Fig. A.34 Maximum number of local search iterations: a) NAHGA, b) SAHGA**

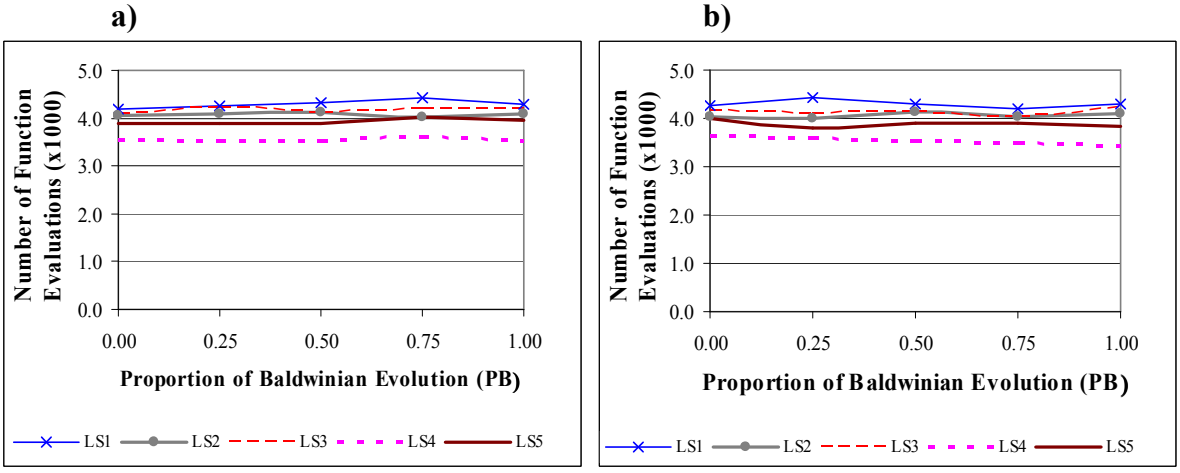



**Fig. A.35 Proportion of Baldwinian evolution: a) NAHGA, b) SAHGA**

# APPENDIX B

# RESULTS FOR TEST FUNCTIONS FROM CHAPTER 3 (ENHANCED SAHGA ALGORITHM)

# APPENDIX B

## APPLICATION OF THE ENHANCED SAHGA ALGORITHM TO TEST FUNCTIONS

This Appendix presents the results for the application of the enhanced SAHGA algorithm presented in Section 5.2. The results are presented for the 8 test problems included in Chapter 3:

- DJ1
- DJ2
- Branin
- Six-Hump
- Schwefel
- Test08
- B&M

Table B.1 details the average savings (relative to the SGA) from applying the enhanced SAHGA algorithm to each of these algorithms. Table B.2 shows the difference in savings attained with the enhanced algorithm and the original SAHGA algorithm.

**Table B.1 Average Savings for Enhanced SAHGA**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 28.5 | 28.2 | 28.2 | 23.7 | 18.0 |
| **DJ2** | 24.6 | 33.4 | 54.3 | 49.4 | 43.6 |
| **Branin** | 43.7 | 45.9 | 51.5 | 47.8 | 47.3 |
| **Six-Hump** | 43.9 | 44.1 | 44.1 | 42.3 | 42.1 |
| **Schwefel** | 34.0 | 37.5 | 39.3 | 35.1 | 30.6 |
| **Griewank** | 22.6 | 24.5 | 24.2 | 20.9 | 24.8 |
| **Test08** | 34.3 | 33.6 | 32.3 | 32.8 | 32.0 |
| **B&M** | 21.5 | 23.0 | 26.1 | 37.0 | 34.5 |

**Table B.2 Difference Between Enhanced SAHGA and SAHGA**

| | Local Search Algorithm | | | | |
|---|---|---|---|---|---|
| **Test Function** | **LS1** | **LS2** | **LS3** | **LS4** | **LS5** |
| **DJ1** | 16.3 | 15.1 | 15.1 | 5.8 | 242.9 |
| **DJ2** | 39.8 | 89.8 | 208.5 | 67.5 | 73.7 |
| **Branin** | 37.4 | 20.8 | 32.7 | 2.4 | 4.6 |
| **Six-Hump** | 47.3 | 48.0 | 48.0 | 44.4 | 44.7 |
| **Schwefel** | 282.0 | 260.6 | 178.9 | 11.4 | 44.3 |
| **Griewank** | 218.3 | 100.8 | 154.3 | 148.8 | 30.5 |
| **Test08** | 41.7 | 23.5 | 14.1 | 44.5 | 16.8 |
| **B&M** | 159.0 | 71.6 | 100.8 | 8.2 | 3.9 |

# Vita

Felipe Espinoza received his Bachelor of Science in Civil Engineering at the University of Chile in Santiago (1989). Two years later he graduated as Civil Engineer with honors. From 1989 to 1997 he worked as a part-time lecturer at the Department of Civil Engineering of the University of Chile. Simultaneously, he worked for the Chilean government and later for the consulting company Ayala, Cabrera and Associates. In January 1998, he was admitted to the Master program in Civil and Environmental Engineering at the University of Illinois at Urbana-Champaign. Later, in January 200 he continued his studies towards the Ph.D. level. In October 2003, he will start a postdoctoral position in groundwater remediation design funded by the National Research Council at the National Risk Management Research Laboratory (NRMRL) depending from the Environmental Protection Agency (EPA Cincinnati, Ohio). His interests include numerical modeling, groundwater remediation design, optimization of groundwater systems, systems analysis and risk analysis.