UNIVERSITY OF CALIFORNIA, SAN DIEGO

Adaptive Global Optimization with Local Search

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science & Engineering

by

William Eugene Hart

Committee in charge:

Professor Richard K. Belew, Chair
Professor Philip E. Gill
Professor Paul R. Kube
Professor Christos H. Papadimitriou
Professor J. Benjamin Rosen
Professor Halbert L. White, Jr.

1994

The dissertation of William Eugene Hart is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

1994

To Valerie

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am indebted to Rik Belew, my advisor. His critical advice and enthusiasm have taught me the balance between careful research and exploratory investigation. I am particularly thankful for his patience and encouragement, which has allowed me to pursue what has interested me.

I am grateful to Paul Kube for encouraging me to work on the STARE project. I am also thankful for his collaborating on various related projects and for his feedback, particularly his critiques of my mathematical analyses. I also thank Scott Baden and Fran Berman for their advice and for inviting me to talk at the parallel computing research group. Each has provided valuable feedback on the parallel aspects of this research.

I thank Mike Goldbaum for providing funding for me on the STARE project throughout most of my dissertation research. I am also thankful to Brad Cote for his cheerful collaboration on the STARE project.

I thank John McInerney for the many discussions we had concerning his dissertation research. These discussions inspired many of the seminal ideas of my research. In many ways, my dissertation is an extension of the John's research.

I also thank Dave Fogel for my "conversations with the devil." He carefully reminded me that other evolutionary algorithms that are not so different from genetic algorithms. Further, his critical questions helped clarify the assumptions that my dissertation research makes about methods of evolutionary computation.

I thank Tim Bailey, Tom Kammeyer, Scott Kohn and Filippo Menczer for collaborating on various research projects. These collaborations were among the most interesting projects I have worked on at UCSD.

Thanks go to the members of the Cognitive and Computer Science Research Group for their discussions, feedback and good times. I am particularly grateful for their patience wading through the mathematical formulas that wormed their way into so many of my talks. Special thanks to Fil for the extra effort he has made reviewing drafts of my papers. Thanks also to Amy Steier for providing construction paper for the ILP posters.

The staff of CSE and csehelp@cs have been very helpful during my stay at UCSD. I am particularly grateful to Kathy Hay for making a special effort to see that my funding was properly managed.

I owe a debt of gratitude to my wife Valerie who traveled across the country so I could come to UCSD. Her friendship, love and encouragement have been unfailing through the highs and lows of graduate school. Finally, I thank Amanda for being born *after* my thesis defense, which made finishing my dissertation much simpler.

VITA

August 13, 1967	Born, Dearborn, Michigan
1989	B.A., University of Michigan, Ann Arbor
1991	M.S., University of California, San Diego
1994	Doctor of Philosophy University of California, San Diego

PUBLICATIONS

William E. Hart and Richard K. Belew. "Optimization with genetic algorithm hybrids that use local search." In *Plastic Individuals in Evolving Populations*, 1994. (to appear).

William E. Hart, Brad Côté, Paul Kube, Michael H. Goldbaum, and Mark R. Nelson. "Automatic segmentation and classification of objects in retinal" images. *IEEE Trans. on Medical Imaging*, 1994. (to appear).

William E. Hart, Thomas E. Kammeyer, and Richard K. Belew. "The role of development in genetic algorithms." In *Foundations of Genetic Algorithms*, 1994. (to appear).

Michael H. Goldbaum, Valentina Kouznetsova, Brad Côté, William E. Hart, and Mark Nelson. "Automated registration of digital ocular fundus images for comparison of lesions." In Jean-Marie Parel and Qiushi Ren, editors, *Proceedings of ophthalmic technologies III*, pages 94–99. SPIE, 1993.

William E. Hart and Richard K. Belew. "Optimizing an arbitrary function is hard for the genetic algorithm." In *Proceedings of the 4th conference on Genetic Algorithms*, pages 190–195, June 1991.

ABSTRACTS

Michael H. Goldbaum, William Hart, and Brad Côté. Automated measures of retinal blood vessel tortuosity. In *Invest Ophthalmol Vis Sci*, 1994. (to appear).

A.W. Dreher, Michael H. Goldbaum, Brad Côté, William E. Hart, and E.D. Bailey. Neural network classification of glaucoma using peripapillary nerve fiber thickness measured by scanning laser tomography. In *Invest Ophthalmol Vis Sci*, volume 34, 1993. #309.

Michael H. Goldbaum, K. Kouznetsova, Brad Côté, L. S. Kirsch, William E. Hart, and Mark Nelson. Automated comparison of lesions in ocular fundus images. In *Invest Ophthalmol Vis Sci*, volume 34, 1993. #1185.

W.S. Karatowski, Michael H. Goldbaum, Brad Côté, William E. Hart, P.A. Sample, and R.N. Weinreb. Screening for glaucoma with few visual field positions selected as optimal by neural networks. In *Invest Ophthalmol Vis Sci*, volume 34, 1993. #1438.

Michael H. Goldbaum, Brad Côté, R. F. Garcia, William E. Hart, Paul Kube, and Mark Nelson. Computer detection and classification of lesions found in diabetic retinopathy. In *Invest Ophthalmol Vis Sci*, volume 33, 1992. #1082.

TECHNICAL REPORTS AND MISCELLANEOUS

William E. Hart and Michael H. Goldbaum. "Registering retinal images using automatically selected control point pairs." In *First IEEE International Conference on Image Processing*, 1994. (submitted).

William E. Hart, Michael Goldbaum, Brad Côté, Paul Kube, and Mark R. Nelson. "Tortuosity measures for blood vessel segments." *IEEE Trans on Medical Imaging*, 1994. (submitted).

Brad Côté, William E. Hart, Michael Goldbaum, Paul Kube, and Mark R. Nelson. "Classification of blood vessels in images of the ocular fundus." Technical Report CS94-350, University of California, San Diego, 1994.

William E. Hart and Richard K. Belew. "Optimization using genetic algorithms with neural network learning rules." In *Proceedings of the INC Annual Research Symposium 3*, Univ. of California, San Diego, 1993.

FIELDS OF STUDY

Major Field: Computer Science

Studies in Genetic Algorithms.
Professor Richard Belew

Studies in Pattern Recognition and Neural Networks.
Professor Paul Kube

Studies in Parallel Algorithms.
Professor Scott Baden

ABSTRACT OF THE DISSERTATION

Adaptive Global Optimization with Local Search

by

William Eugene Hart
Doctor of Philosophy in Computer Science & Engineering
University of California, San Diego, 1994
Professor Richard K. Belew, Chair

This dissertation examines the performance of genetic algorithm (GA) hybrids that use local search to solve global optimization problems. GAs are a class of adaptive global sampling methods that take many cues from mechanisms observed in natural evolution. GAs maintain a population of solutions that are used to generate new solutions in the search space. GA hybrids using local search (GA-LS hybrids) are motivated by the apparent need to employ both a global and local search strategy to provide an effective global optimization method. Previous experimental results have found that GA-LS hybrids not only find better solutions than the GA, but also optimize more efficiently.

To improve the efficiency of GA-LS hybrids, I propose and experimentally validate methods that selectively apply local search to solutions in the GA's population. First, local search is randomly applied with a fixed frequency. Experiments with this method illustrate a trade-off between the refinement performed by local search and the reliability of the competitive search performed by the GA. Next, I describe two classes of adaptive methods. *Distribution-based* adaptive methods use redundancy in the population to avoid performing unnecessary local searches. *Fitness-based adaptive* methods use the fitness information in the population to bias the local search towards individuals that have better fitness. An experimental analysis indicates that these adaptive methods can significantly improve the efficiency of GA-LS hybrids.

This dissertation explores implications of these results for parallel GAs. In particular, a MIMD design for geographically structured genetic algorithms (GSGAs) is described. GSGAs were initially developed for SIMD architectures, where it is difficult to selectively apply local search. An analytic and experimental analysis of MIMD GSGAs demonstrates that they scale well for large problems.

GA-LS hybrids are used to solve global optimization problems in several application domains. First, GA-LS hybrids are used to find the weights of a neural network to solve the six-bit symmetry problem. Next, they are used to solve a simple 19 atom molecular conformation problem. Finally, they are applied to a drug docking problem. When compared to simulated annealing, GA-LS hybrids not only find better solutions and optimize more efficiently.

Chapter I

Introduction

I.A Global Optimization

Many practical engineering problems can be formulated as optimization problems using an objective function whose domain, D, represents the space of feasible solutions (points) and whose range represents the relative utility of each solution. Solving the optimization problem requires the computation of the global minima or maxima of the objective function. Without loss of generality, assume that the objective function is minimized and that there is a unique global minimum. This dissertation examines objective functions of the form $f : D \rightarrow \mathbf{R}$, $D \subseteq \mathbf{R}$. The aim of global optimization is to find x^* such that

$$x^* = \arg\min_{x \in D} f(x).$$

In practice, we need to account for the fact that numerical procedures can only produce approximate answers. Hence we consider the problem solved if for some $\epsilon > 0$ we find a solution in the *level set* L_d, where $d = f(x^*) + \epsilon$ and

$$L_d = \{x \in D \mid f(x) \leq d\}.$$

We say that a solution $x \in L_d$ is ϵ-accurate.

Powerful local search techniques have been developed to solve optimization problems. A local search algorithm is one that iteratively improves its estimate of the minimum by searching for better solutions in a local neighborhood of the current solution. The neighborhood of a local search algorithm is the set of solutions that can be reached from the current solution in a single iteration of the local search algorithm. Local search techniques have been developed for which stopping conditions can terminate the search at a local minimum of an objective function. If $nbhd(x)$ is a local neighborhood of x, then x is a local minimum if $f(x) = \min\{f(y) \mid y \in nbhd(x)\}$.

In general, local minima are not guaranteed to be global minima. Consequently, global optimization methods have been developed to perform a sophisticated search across multiple local minima. Global optimization is an inherently difficult problem since no general criterion exists for determining whether the global optimum has been reached.

I.B Adaptive Global Optimization

Törn and Žilinskas [91] observe that two competing goals govern the design of global optimization methods. *Global reliability* is needed to ensure that every part of the domain is searched enough to provide a reliable estimate of the global optimum. *Local refinement* is important since the refinement of the current solution will often produce a better solution. Most global optimization algorithms achieve these two goals using a combination of a global strategy and local strategy.

This dissertation focuses on global optimization methods that combine adaptive global sampling methods with local search. Adaptive global sampling methods vary the sampling distribution depending upon the objective function's values on previously sampled solutions. This adaptation usually biases the sampling towards regions of the search space where near-optimal solutions have been discovered. Genetic algorithms (GAs) are an interesting class of adaptive global sampling methods that take many cues from mechanisms observed in natural evolution. GAs maintain a population of solutions that are used to generate new solutions in the search space. They adapt their global sampling by performing a competition between solutions that selects better solutions with greater frequency. The competitive selection at each iteration (generation) of a GA biases the sampling performed in subsequent generations, thereby adapting the global sampling.

Törn and Žilinskas' observation suggests that when a GA is used as a global function optimizer, its standard operators be augmented with the ability to perform local search. GA hybrids that use local search (GA-LS hybrids) can use local search in one of two ways. GAs and local search can be applied in separate phases, using local search to refine solutions generated by a complete run of the GA. Alternatively, local search can be applied to solutions in each iteration of the GA. This type of GA-LS hybrid is particularly interesting because the global and local search methods can influence each other's behavior. An important example of this phenomenon is the Baldwin effect [6, 40] in which learning in natural systems speeds up the rate of evolutionary change. Similar effects have been observed by a number of authors using GA-LS hybrids [7, 40, 50]. The research in this dissertation examines the second type of GA-LS hybrid.

I.C Genetic Algorithms with Local Search

Previous experimental results confirm that GA-LS hybrids not only find better solutions than the GA, but also optimize more efficiently [7, 61]. It is noteworthy that these results examine a limited number of algorithmic combinations of the GA with local search. I believe that important algorithmic combinations have been overlooked and that the standard GA-LS hybrids of the GA and local search should be reconsidered.

In this dissertation, I propose and experimentally validate several nonstandard GA-LS hybrids. The following issues have motivated the algorithmic combinations that are examined.

I. How often should local search be applied? Standard GA-LS hybrids apply local search to every individual in the GA's population. While this design makes full use of the potential information provided by the local search, the cost of the local search method places constraints on the GA-LS hybrids. For example, the cost of the local search method has constrained many researchers to use small populations sizes in order to allow the GA to run multiple generations. Further, applying local search to every individual is does not necessarily improve the efficiency of the GA-LS hybrid's search since local searches may be done on solutions which are clearly not near the global optimum. I propose methods for which the frequency of local search is automatically adapted, and describe how the optimal local search frequency is related to the type of problem being optimized.

II. On which solutions should local search be used? If local search is not applied to every individual in a population, then we need to decide how individuals are selected for local search. The simplest method of selecting individuals is uniformly at random. I describe more sophisticated methods that use information from the population to bias the selection of individuals for local search. First, I describe methods that use the redundancy of solutions in the population to modify the rate at which local search is applied to each individual. These methods reduce the chance that an individual will be used for local search if that solution is similar to other individuals in the population. I also describe methods that use the values of solutions in the population to select individuals for local search that are more optimal.

III. How long should the local search be run? The basin of attraction of a local minimum is the set of solutions from which local search will converge to that local minimum. When using local search in the basin of attraction of the global optimum, one clearly wants to perform a complete minimization to the global optimum. However, when performing local search in other basins of attraction, complete minimization may not be necessary. If the GA-LS does not require refined solutions to discriminate between two regions of the search space, then complete minimization may not be necessary. I examine this issue by comparing the efficiency of GA-LS hybrids using several local search lengths.

IV. How efficient does the local search need to be? It is often possible to have several local search methods available for a particular search domain. For example, when optimizing smooth functions on $\mathbf{R}^n$, numerous local search methods have been proposed using derivative information to perform the local search. One is often faced with a choice between two or more local search methods with different efficiencies, such that the more efficient local search algorithms are more expensive to run. When selecting a local search method for a GA-LS hybrid, the cost of the local search and its efficiency are both factors that may affect the overall efficiency of the hybrid. To evaluate the effect of this trade-off, I examine GA-LS hybrids that apply local search algorithms which use different types of derivative information.

Taken together, issues I and II pertain to the manner in which local search is *selectively applied* to the GA's population. These issues are the central focus of the experimental analysis of GA-LS hybrids, and factors relating to the remaining issues are considered as part of this analysis. Our approach is to improve these GA-LS hybrids by introducing mechanisms to selectively apply local search within each generation. These mechanisms provide a general means for reducing the number of local searches that can be used with a wide variety of optimization problems. My thesis is that selectively applying local search can improve the efficiency of each iteration of the GA while preserving the benefits of the hybridization.

I.D Parallel Genetic Algorithms with Local Search

Research on GA-LS hybrids has been performed on both sequential and parallel architectures (see McInerney [56] for a review of parallel GAs). Parallel GAs have been motivated by the need to process large populations when solving high dimensional problems. They are also important when solving problems for which the objective function is expensive to evaluate.

Most of the research on parallel GA-LS hybrids has been performed with coarse-grained MIMD architectures. These computers offer parallelism among a limited number of processors that run asynchronously. McInerney [56] has also analyzed GA-LS hybrids on a fine-grained SIMD architecture, the CM-200. This computer offers parallelism among a very large number of processors that execute each instruction synchronously. Asynchronism is particularly important because GA-LS hybrids are naturally asynchronous. The time needed to perform a local search can vary depending on the starting point, and the local search algorithm itself may not be amenable to a SIMD parallelization. To account for these constraints, McInerney's SIMD GA-LS hybrid uses a truncated local search in which all individuals in the population take a few steps in the gradient direction.

I propose and analyze a MIMD design for geographically structured genetic algorithms (GSGAs). The SIMD GAs examined by McInerney and others are called GSGAs because they spatially structure the adaptive search performed by the GA. Gordon and Whitley [35] have recently argued that the algorithmic nature of GSGAs may be of interest independent of their implementation on a particular architecture and observe that their performance is competitive with other parallel GAs.[1] An analytic and experimental analysis of MIMD GSGAs demonstrates that they scale well for large problems.

I.E Dissertation Overview

The issues and contributions outlined in the previous sections are elaborated in the following dissertation chapters. Chapter II presents background material in local search, global optimization and genetic algorithms. It also discusses previous work that is related to the research presented in this dissertation.

Chapter III presents a simple extension of the multistart algorithm, which performs local search using a nonadaptive global sampling method. The new algorithm uses local search with a fixed frequency, which can be viewed as a simple form of selecting points to perform local search. This analysis indicates that for any given

[1] For technical reasons, GSGAs are called Cellular GAs by Gordon and Whitley.

function, it is always more efficient to perform local search with frequency of either zero or one. Thus, this simple form of selecting local search does not improve the multistart algorithm.

Chapter IV reviews my previous research that analyzes the complexity of the optimization problem for the GA. I conclude that the space of possible functions is an important aspect of any analysis of the GA's performance, which leads us to consider a set of test functions on $\mathbf{R}^n$. Finally, I describe a GA that uses a floating point representation.

Several methods of selectively applying local search in GA-LS hybrids are proposed in Chapter V. The first simply uses a fixed frequency of local search, but provides considerable insight into the role that local search plays. Next I propose methods that use the redundancy in the population to reduce the local search frequency. Finally, I propose methods that use the population's fitness information to bias the selection towards more optimal individuals.

Chapter VI describes a MIMD design for GSGAs. An analysis of the algorithm's efficiency is performed, which is extended to GSGAs that use local search with a fixed frequency. Experimental results closely match the predictions of the analysis, and exhibit good scaling properties.

Chapter VII describes the application of these methods to neural networks, a simple conformation problem and a drug docking problem.

In Chapter VIII, I summarize my findings, discuss implications for related research with natural evolutionary systems, and point to future research directions.

Appendix A describes a generalization of the biological notion of F statistics. Appendix B provides formulas for the analytic gradients of the simple conformation problem discussed in Chapter VII.

Chapter II

Background and Related Work

This dissertation has been influenced by a number of different fields. The goal of this chapter is to review literature from these different fields and discuss related work.

I begin by providing an overview of local optimization and describe several local search algorithms that will be used throughout the dissertation. Next I discuss the global optimization problem and review standard methods of global optimization. This review highlights the use of local search in these methods. It also distinguishes adaptive and non-adaptive methods of global search. The GA is identified within this context, and is described in more detail. Geographically structured GAs are described and contrasted with standard, panmictic GAs. Next I describe how GAs can be hybridized with local search algorithms. Finally, I discuss previous work that is related to the research presented in this dissertation.

II.A Local Search

Methods of local search have gained attention in both theoretical computer science and numerical optimization. An important distinction among local search methods concerns whether they minimize in the presence of constraints that restrict the domain of the search [26]. This dissertation examines methods for unconstrained optimization.

Theoretical computer science is primarily interested in local search methods over discrete spaces. Johnson, Papadimitriou and Yannakakis [48] observe that "One of the few general approaches to difficult combinatorial optimization problems that has met with empirical success is *local* (or *neighborhood*) *search*." For example, local search methods have proven very successful for the celebrated Traveling Salesman problem [47].

A number of authors have performed general analyses of local search methods over discrete spaces. Tovey [92, 93] models the expected performance of local search algorithms that optimize real valued functions defined on $\{0,1\}^n$. Johnson, Papadimitriou and Yannakakis [48] introduce the complexity class PLS (Polynomial Local Search). Members of PLS are problems for which a local minimum can be found using a polynomial-time local algorithm that finds a solution with better cost, or identifies the current solution as a local optimum. Papadimitriou, Schäffer and Yannankakis [71] use this class of problems to show how local search is the main underlying method used to solve seeming unrelated problems in computer science.

The field of applied mathematics is primarily interested in local search methods used for minimizing continuous functions on compact spaces. An important distinction among these methods concerns the use of derivative information like gradients, $f'(x)$, and Hessians, $f''(x)$; algorithms can be distinguished by the highest order derivatives that they use. Algorithms using derivative information of order greater than zero are somewhat more powerful than those which only use function evaluations (order zero derivatives). However, derivative information requires additional calculations, and these algorithms do not always generated good solutions fast enough to compensate for the additional expense.

Three methods of local search will be used throughout this dissertation. The first is the non-derivative method proposed by Solis and Wets [84]. Next, conjugate gradient methods [26, 74] are used to minimize continuous functions using gradient information. Finally, stochastic approximation is used in pattern recognition methods to find the optimal weights for parametric models of data [19].

II.A.1 Random Local Search

Solis and Wets [84] propose several random local search methods for performing local search on smooth functions without derivative information. Their "Algorithm 1" uses normally distributed steps to generate new points in the search space. A new point is generated by adding zero mean normal deviates to every dimension of the current point. If the value of the new point is worse than the current point, then this algorithm examines the point generated by taking a step in the opposite direction from the new point. If neither point is better than the current point, another new point is generated.

This algorithm depends upon parameters that automatically reduce and increase the variance of the normal deviates in response to the rate at which better solutions are found. If new solutions are better sufficiently often, the variance is increased to allow the algorithm to take larger steps. If poorer solutions are frequently generated, the variance is decreased to focus the search near the current solution.

Unfortunately, this algorithm does not have well defined stopping conditions. Solis and Wets examine several attempts to define stopping criteria for random search techniques, and conclude that "... the search for a good stopping criterion seems doomed to fail." In practice, this method is halted after a fixed number of iterations, or when the step size becomes smaller than a given threshold.

II.A.2 Conjugate Gradient

Several classes of local search algorithms have been defined for algorithms that use gradient information. Among them, conjugate gradient methods provide an efficient use of the gradient information while only requiring $O(n)$ storage [74].

Conjugate gradient methods are motivated by an analysis of the steepest descent method. The steepest descent method iteratively performs line searches in

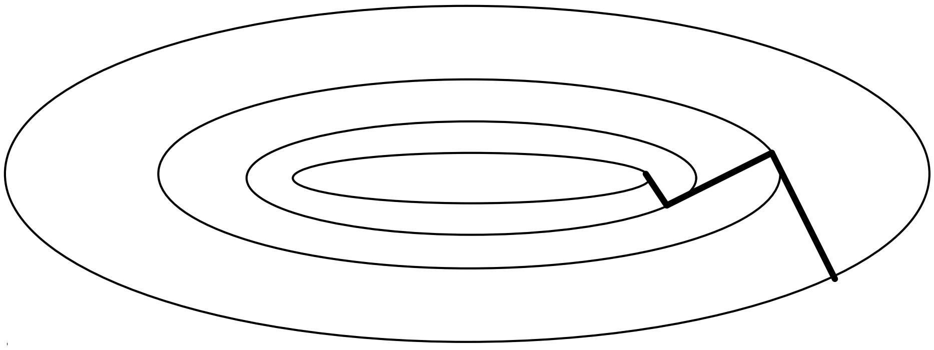


Figure II.1: Performance of the steepest descent method on a narrow valley.

the local downhill gradient direction $-\bigtriangledown f(x)$. A line search performs a minimization through a one dimensional slice of a function, specified by an initial search direction. Thus, the steepest descent method iteratively minimizes the objective function in the gradient direction.

Consider the path of the steepest descent method shown in Figure II.1. When applied to functions like this that have narrow "valleys", the steepest descent method is inefficient. You might expect that the first line minimization would take you to the bottom of the valley, and the second would finish the minimization. However, the new gradient at the minimum of the first line search is perpendicular to the first gradient. Thus the new gradient does not, in general, point toward the local minimum.

Conjugate gradient methods remedy this situation by using search directions that are conjugate to the previous search directions (the initial search direction is the downhill gradient). The notion of conjugacy attempts to preserve the minimization along the previous search directions by requiring that the change along the current gradient remain perpendicular to the previous search directions. A quadratic function can be expressed as

$$f(x) = c + b^T x + \frac{1}{2} x^T A x,$$

where c and b are vectors and A is symmetric. For quadratic functions, using conjugate search directions guarantees that subsequent line searches preserve the previous minimizations. Hence, $O(n)$ line searches are needed [74].

Because it uses gradient information, conjugate gradient has well-defined stopping criterion. The conjugate gradient method uses gradient information do terminate when the algorithm has reached a critical point of the objective function [26, 74].

II.A.3 Stochastic Approximation

In pattern recognition problems, one is often given a set of data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ for which we would like to know the relationship between the x_i and the y_i. One common approach to this problem is to propose a parametric model $f(x, w)$ and use minimization techniques to determine parameters w that minimize

$$J(w) = \sum_{i=1}^{n} E(y_i, f(x_i, w)),$$

where $E(\cdot, \cdot)$ computes the error between the predicted y value and the actual y value, y_i. A common error function is the squared error

$$E(a, b) = \|a - b\|^2.$$

Examples of parametric models are linear models [19], logit models [13] and neural networks [81].

Both random local search and conjugate gradient methods can be used to minimize $J(w)$, since gradient information is typically available for this function. An alternative method of minimizing $J(w)$ is *stochastic approximation*. Unlike these other methods, stochastic approximation makes changes to the current solution based on partial calculations of $J(w)$. In particular, it makes updates to the current solution using randomly selected samples.

To use information from a single sample, suppose that (x_i, y_i) is randomly selected from the data set. The following learning rule is described in White [99] and

Rumelhart, Hinton and Williams [81]:

$$w_{t+1} = w_t + \Delta w_t \tag{II.1}$$

$$\Delta w_t = -\eta_t \nabla_w E(y_i, f(x_i, w_t)), \tag{II.2}$$

where η_t is the so called *learning rate*, which controls the step size of this method. Rumelhart, Hinton and Williams call this the *back-propagation* learning rule and discuss the following extension

$$\Delta w_t = -\eta_t \nabla_w E(y_i, f(x_i, w_t)) + \alpha \Delta w_{t-1}, \tag{II.3}$$

where α is the so called *momentum rate*, which is used to retain a "memory" of previous steps. White [99] summarizes analyses of stochastic approximation methods which show conditions under which w_t converges to the optimum with probability one.

II.B Global Optimization

Methods of global optimization differ from methods of local optimization in that they attempt to find not just any local optimum, but the smallest (largest) local optimum in the search space D. Global optimization problems are inherently difficult, and few assumptions can be made about problems of practical interest. The methods described below assume that the function is almost everywhere continuous over D. In general, methods that utilize *a priori* information about a problem will outperform general purpose methods that utilize less information. However, in many practical problems information beyond these basic assumptions will be unavailable.

One important characteristic of global optimization methods concerns whether their estimate of the global optimum is guaranteed to converge to the global optimum. For a deterministic algorithm, the estimates of the global optimum, x_n, converge if $\lim_{n\to\infty} x_n = x^*$. Natural generalizations of convergence can be defined for stochastic algorithms [99]. Unfortunately, convergence is typically provided in a limit that is

I. Methods with guaranteed accuracy

   A. Covering methods

II. Indirect methods

   A. Methods approximating the level sets

   B. Methods approximating the function

III. Direct Methods

   A. Clustering methods

   B. Generalized descent methods

   C. Random search methods

Figure II.2: Classification of global optimization methods.

unattainable in practical terms. Time constraints typically preclude the ability to search enough to guarantee convergence to the optimum, so heuristics are often used to generate near-optimal solutions rapidly.

I now review standard methods of global optimization. Because my interest concerns adaptive global search methods that use local search, I pay close attention to the role of local search techniques in these global optimization algorithms. Figure II.2 shows the classification of global optimization methods proposed by Törn and Žilinskas [91] (our category labels).

II.B.1 Methods with Guaranteed Accuracy

The covering methods use a global search strategy that excludes regions of the search space based on estimates of how much the function can vary over small regions. For example, quasi-Monte Carlo methods [65, 66, 67] deterministically generate a sequence of points that are uniformly spread across the search space. The accuracy of the estimated global optimum is computed using measures of the uniformity of the sequence of points. Covering methods do not usually incorporate local search strategies, though they could refine their final estimate by performing local search with the best solution found. While covering methods have provable convergence properties, they generally require the user to estimate properties like the Lipschitz constant. Unfortunately, these properties can be difficult to estimate, so the utility of these algorithms is unclear in many practical applications [76].

II.B.2 Indirect Methods

Indirect methods use local information like function evaluations to build a model of either the function or its levels sets. This model is then used to guide the selection of new samples. Since the construction and maintenance of the model of the function can be quite costly, these approaches are appropriate when the objective function is very expensive to evaluate. Neither of the indirect approaches mentioned above use local search strategies. According to Törn and Žilinskas, these methods are especially useful in single dimensional problems, though they have been successfully applied to problems with dimensionality less than or equal to 15.

II.B.3 Direct Methods

The algorithms that we examine in this dissertation can most naturally be classified as direct methods. Direct methods differ from indirect methods in that they do not perform expensive processing on the local information. Instead, they directly use the local information itself to guide the global and local search.

Generalized Descent

The methods for generalized descent attempt to retain the basic functionality of the standard local search procedures while performing global search. *Trajectory methods* modify the trajectory of the local search routine so it passes through all of the local optima. For example, the method proposed by Fiodorova [20] is composed of three subalgorithms that are used to (1) descend toward a local minimum, (2) ascend from a minimum up to a saddle point, and (3) pass through a saddle point. Using these subalgorithms, new local minima are identified from searches originating from previously identified local minima. *Penalty methods* modify the objective function with penalty terms that make the local search procedure avoid the local minima that it has previously searched. The *tunneling method* described by Gomez and Levy [32] uses two phases: local minimization and tunneling. The local minimization phase finds a local minimum x'. The tunneling phase minimizes a modified objective function to find a point x'' such that $f(x'') < f(x')$. The modified objective function is designed such that a local search procedure can be used to search for x'' starting from x'. Törn and Žilinskas [91] note that the implementation of generalized descent techniques is similar to a multistart procedure using non-local optimization techniques (see below).

Clustering Methods

Clustering methods are among the most efficient algorithms proposed for global optimization. These methods are composed of several steps. First, they perform Monte Carlo sampling of the search space. The samples are concentrated to obtain groups around the local minima and then clustered to give clusters identifying local minima. Finally, a complete local search is applied to a sample from each cluster. A variety of methods can be used to perform each of these steps. Concentrating the samples is typically performed by refining the samples with a few steps of local search and retaining a fraction of the best samples. Törn and Žilinskas [91] describe a number of clustering algorithms that have been used with these methods, including standard hierarchical methods. Clustering methods are amenable to analysis because they use uniformly distributed samples. Rinnooy Kan and Timmer [77, 78] describe a clustering method and describe conditions for which any local minima will be found within a finite number of iterations with probability one.

One drawback of cluster methods is that they tend to perform poorly on functions with many local minima. For these functions, many more samples are needed to identify the local minima. It is unclear whether the relatively poor performance on these types of functions results from inadequate stopping criteria or from a bias in the clustering methods towards larger clusters. These techniques have been successfully applied to problems with as many as 40 dimensions [73].

Random Search

In Törn and Žilinskas, the category of random search methods is a collection of techniques that use randomization and which do not fit nicely into any of the other categories. I identify two subclasses of random search methods that are relevant to this dissertation: IIIC.1 blind random search and IIIC.2 adaptive random search.

Blind Random Search *Blind* random search methods use a global search strategy that does not use information from previous samples to guide the selection of the current sample. Because these methods ignore previous samples, they may also be called *non-adaptive*. The Monte Carlo and multistart algorithms are examples of this type of algorithm. The Monte Carlo algorithm (MC) randomly samples from the search space according to a fixed distribution. The multistart algorithm (MS) uses MC to generate samples on which local search is performed. MS is a blind random search method, because it uses MC to generate global samples.

Variants of these algorithms that use a fixed, non-uniform distribution over the search domain are also blind random search techniques. Also included is the probabilistic multistart algorithm (described in Chapter III) and the algorithm described in Dorea [17] for which new samples are generated by adding a random deviate (from a fixed distribution) to the previous sample. Blind random search methods are relatively inefficient, but they are often amenable to analysis [8, 18]. These methods are limited in their use of local search because the global sampling method is not influenced by the performance of the local search algorithm.

Adaptive Random Search *Adaptive* random search methods differ from blind random search methods by using information from previous samples to guide the selection of the current sample. Examples of adaptive random search methods considered in this thesis are simulated annealing and evolutionary algorithms.

Simulated annealing (SA) is a method of optimization inspired by an analogy between a physical annealing process for obtaining low energy states and the process of solving for minimal solutions to discrete optimization problems [11, 51]. SA sequentially generates random deviates of the current solution that are accepted if a probabilistic test is passed. Suppose x' is the current solution and let x'' be the new deviate. If $f(x'')-f(x') < 0$, the new deviate is accepted. Otherwise, the deviate is accepted with probability

$$\exp^{(f(x'')-f(x'))/T} .$$

The value T is the "temperature" parameter that is annealed during the course of the optimization. Initially, the probability of acceptance is high, and eventually it becomes small. While SA is used for global optimization, it makes no clear transition between performing global and local search. At high temperatures, it will frequently make uphill moves, which enable it to perform a global search. As the temperature decreases, the search becomes increasingly localized. At very low temperatures, the search is often localized to a single basin of attraction for which there is a low probability of escaping in the near term. For this reason, simulated re-annealing has been proposed [44, 45]. This variant treats simulated annealing more like a local search technique, using multiple starts to perform the global search.

II.C Evolutionary Search

Evolutionary search algorithms, called *competitive* search by Törn and Žilinskas [91], represent an important class of adaptive search algorithms. Evolutionary search is an adaptive random search that maintains a collection of solutions that are ranked by their performance and uses a competition between these solutions

Initialize population (with uniformly generated solutions)
Repeat
    Evaluate solutions in the population
    Perform competitive selection
    Apply genetic operators
    Perform local search (optional)
Until convergence criteria satisfied

Figure II.3: Pseudo-algorithm for a genetic algorithm.

to select solutions for further processing. Research on evolutionary search algorithms incorporates elements of both biological evolution and global optimization. These algorithms are inspired by biological evolutionary mechanisms and are often used to perform global optimization.

The exemplars of evolutionary search algorithms are genetic algorithms, evolutionary strategie and evolutionary programming [5, 22, 31]. The design and motivation for these algorithms are different, but they incorporate the same basic adaptive components [4, 41]. These methods use a collection of solutions (population of individuals) that are updated iteratively using selection mechanisms and genetic operators. The general process of each iteration (generation) is described in figure II.3.

The selection mechanism performs a competition to select a subset of the solutions for further processing. The genetic operators are used to generate deviates from the selected individuals. Two types of genetic operators are commonly employed: mutation and recombination. Mutation uses a single individual to generate a deviate that is located in the local neighborhood of the individual. Recombination uses two individuals to generate another individual that is typically located in the smallest hypercube that contains them both. Local search is another genetic operator that is sometimes employed with GAs to refine solutions in their local neighborhood.

Using these genetic operators, evolutionary search algorithms perform a global search. Global convergence is not guaranteed for all evolutionary algorithms [79], but experiments with these algorithms indicate that they often converge to regions of the search space that contain near-optimal solutions. Global convergence is guaranteed for the type of GAs used in this dissertation.

Local search is particularly interesting in the context of GAs because the recombination operator may prove a powerful method for adaptively generating solutions in new basins of attraction. Since evolutionary programming uses only the mutation operator to generate new solutions, we expect that it will have greater difficulty generating solutions in new basins of attraction. Evolutionary strategie also uses recombination, so it may be interesting to use local search with this algorithm.

II.C.1 Genetic Algorithms

The GA was initially described using populations of binary strings in $\{0,1\}^n$, which are evaluated by the objective function (fitness function) [42, 31, 57]. When searching spaces other than $\{0,1\}^n$, the objective function decodes the binary string and performs the function evaluation.

Holland [42] proposed a selection mechanism that stochastically selects individuals with probability

$$p_i = \frac{f(x_i)}{\sum_i f(x_i)}$$

This selection mechanism is called *proportional selection*, since the number of copies of an individual will be in proportion to the its fraction of the population's total fitness. This method assumes the GA is minimizing $f(x)$ and that the global minimum is greater than or equal to zero, but it can be easily modified to perform selection when maximizing a function, or when the global minimum is negative.

The binary GA proposed by Holland uses mutation and crossover operators. With binary strings, the mutation operator changes a single bit on a string, and it is typically used with low frequency. The crossover operator picks two points on the the binary representation and generates the new sample by taking all of the bits between these points from one parent and the remaining bits from the other parent.

For example, if $n = 10$ and the chosen points are $p_1 = 2$ and $p_2 = 6$:

```
Parent(1): 1111111111    Parent(2): 0000000000    Sample: 0011110000
```

Crossover is typically used with high frequency, so most of the individuals in each generation are generated using crossover.

The manner in which the parameters of the objective function are encoded on each string does not affect the mechanisms of the GA, though it can affect the GA's search dynamics. In particular, much research has been done examining how crossover composes and disrupts patterns in binary strings, based on their contribution to the total fitness of the individual [30, 85, 86, 97]. This research has motivated the use of modified crossover operators that restrict the distribution of crossover points. For example, if the binary string is decoded into a vector of integers or floating point values, then crossover is often applied only between the integer or floating point values on the binary string [15].

II.C.2 Panmictic and Geographically Structured Genetic Algorithms

GAs can be distinguished by the manner in which the selection mechanism and genetic operators are applied to the population. *Panmictic* GAs use selection mechanisms (like proportional selection) that use global information about the entire population to perform a global selection. In proportional selection the population's total fitness is used to perform selection. Panmictic GAs apply the crossover operator to pairs of individuals randomly taken from individuals selected from the entire population.

Geographically structured genetic algorithms (GSGAs) perform a structured selection in which individuals compete against a fixed subset of the population, and the genetic operators are applied to individuals selected from these subsets. The most common way of structuring the selection mechanism uses a toroidal two dimensional grid like the one in Figure II.4 [2, 12, 56, 87]. Every element of the population is

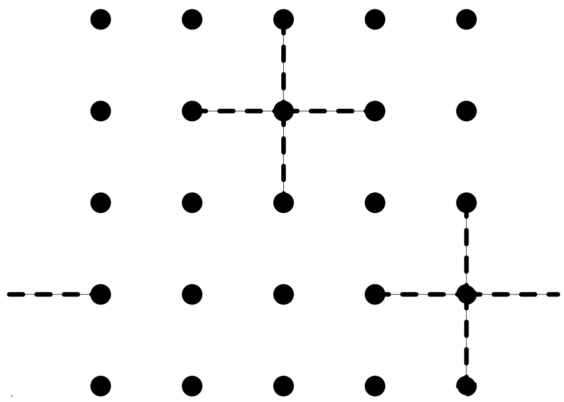


Figure II.4: The two dimensional grid used by GSGAs to define population subsets.

assigned to a location on the grid. The grid locations are not necessarily related to the individuals' solutions. They are often arbitrary designations used to perform selection. Thus, there are distinct notions of locality with respect to the population grid and the search space (see Figure II.5). When local search is performed with GSGAs, it is performed in the search space. When local selection is performed, it is performed in the population grid.

Two general methods of local selection have been used to perform selection in GSGAs: (1) fixed size neighborhoods have been used to define the set of neighboring individuals [14, 35], and (2) random walks have been used to stochastically sample the locations of neighboring individuals [12, 56]. Figure II.4 illustrates the fixed size neighborhoods that could be used to perform selection. Proportional selection is applied to the solutions in each of these neighborhoods. Since one individual is assigned to each grid location, the selection procedure is used to select only as many individuals as are necessary to use the genetic operators. For example, two individuals will be selected if crossover is used. The new individual generated from a genetic operator is assigned to the grid location at which selection is performed.

The early motivation for GSGAs came from SIMD designs for GAs (see Chapter VI). McInerney [56] describes a SIMD GSGA and analyzes the effect of different methods of local selection. He shows how local selection encourages local

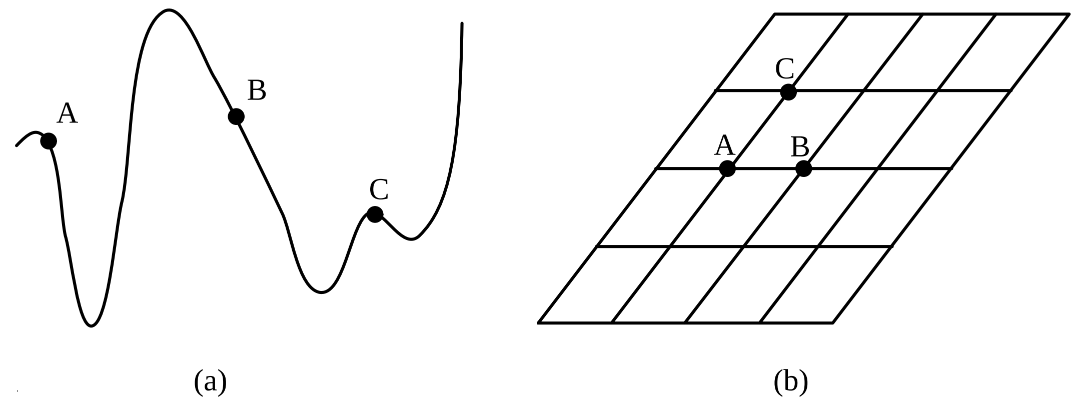


Figure II.5: An illustration of the two notions of locality in GSGAs: (a) locality in the search space, and (b) locality in the two dimensional grid used by GSGAs.

regions of the 2D grid to form *demes* of very similar individuals, and argues that inter-deme competition enables GSGAs to perform search while maintaining diversity in the population. He observes that selection using random walks gave very good results in his experiments. He notes that this method enabled good solutions to diffuse through the population, while strongly encouraging the formation of demes.

Gordon and Whitley [35] have recently argued that the algorithmic nature of GSGAs may be of interest, independent from their implementation on a particular architecture. They experimentally compare GSGAs to panmictic GAs and observe that the GSGAs provide superior performance. This philosophy is echoed by Davidor, Yamada and Nakano [14] in their motivation for the ECO framework. The ECO framework provides a serial design for implementing a geographically structured GA.

Finally, we note that our definition of GSGAs includes GAs which structure the selection at a fine granularity. A number of GAs have been proposed whose competitive selection is intermediate between GSGAs and panmictic GAs. Mühlenbein [63] makes a similar distinction and describes a GA which uses a set of independent subpopulations and structures the inter-population communication with a ladder structure. These subpopulations are typically small, so they perform a localized search of the function. For example, Whitely [102] illustrates how a small population can perform a locallized search in the context of neural network optimization problems. Inter-population communication enables populations to combine disparate solutions and enables them to perform a global search.

II.C.3 GAs with Local Search

GA hybrids that use local search (GA-LS hybrids) are motivated by the apparent need to employ both a global and local search strategy to provide an effective global optimization method. The GA performs an adaptive, global sampling of the search domain, but it does not efficiently refine solutions. GA-LS hybrids use local search to efficiently refine solutions, and provide a clear separation between the global and local search being performed by the algorithm.

The use of local search with GAs is also inspired by biological models of learning and evolution. We have noted that evolutionary algorithms like the GA take many cues from mechanisms observed in natural evolution. Similarly, models of learning are often equated with techniques for local optimization [81]. Research on the interaction between evolution and learning has naturally led computer scientists to consider interactions between evolutionary algorithms and local optimization [7].

The following framework is used to describe the range of interactions between the GA and local search algorithms. Let $\mathcal{G}$ be the space of genotypes, and let $\mathcal{P}h$ be the space of phenotypes. Genotypes are mapped to phenotypes via a maturation function, $\delta : \mathcal{G} \rightarrow \mathcal{P}h$. This is a restricted notion of maturation, since the phenotype is generated only with information that is available in the genotype. Let the function $f(x)$ be the fitness function, $f : \mathcal{P}h \rightarrow \mathbf{R}$.

Local search algorithms employ information about the fitness landscape, so local search is performed in $\mathcal{P}h$. Iterative moves of the local search are defined using a local search operator $\mathcal{L}$:

$$ph_0 = \delta(g), \;\; ph_1 = \mathcal{L}(ph_0), \;\; ph_2 = \mathcal{L}(ph_1), ..., \;\; ph_n = \mathcal{L}(ph_{n-1})$$

Local search operators may exploit any information about the fitness function (e.g., derivatives of f) to estimate a solution with a better fitness value.

This differentiates them from mutation operators, $\mathcal{M}$, which depend only on information contained in g; in particular, they are independent of information about either the phenotypic representation or other individuals in the population. Note that "mutation" is sometimes used to refer to any and all genotypic modifications. We reserve the term "mutation" for completely random, "blind" modifications. Specifically, this notion of mutation does not include modifications like crossover that exploit information (e.g., population gene frequencies) to select the changes made to the genotype.

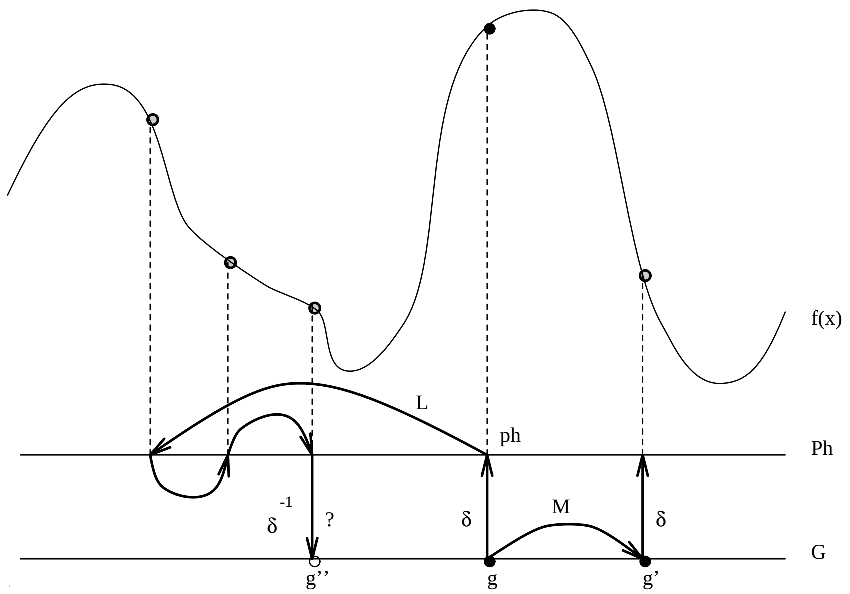


Figure II.6: Illustration of genotypic and phenotypic local search.

Figure II.6 illustrates the interaction between the various elements of the framework that we have described. This figure shows how mutation and local search take a genotype g and generate new genotypes g' and g''. The mutation operator simply generates another genotype, while local search uses the maturation map to generate a phenotype which is modified using fitness information. When local search is used, the genotype g may or may not be modified. *Lamarckian local search* replaces

g with the result $\delta^{-1}(ph_n)$. The name is an allusion to Jean Batiste de Lamarck's contention that (some) phenotypic characteristics acquired during a lifetime can become heritable traits. In our model, acquired characteristics correspond to phenotypic modifications due to the local search operator, and heritability corresponds to the replacement of genome g with $\delta^{-1}(ph_n)$. If $\delta^{-1}(x)$ does not exist, then Lamarckian local search is not possible since the "reverse transcription" of genetic material cannot be performed. In this case, only *non-Lamarckian local search* can be performed. *Non-Lamarckian local search* exploits information gained via phenotypic search without using it to directly modify the genome. Non-Lamarckian local search is typically used to determine the fitness associated with g.

II.D Related Work

GAs have been combined with local search methods for a number of different applications. The problem of finding the optimal parameters for a neural network [7, 50, 68] comes closest to the models of learning and evolution. GA-LS hybrids have been applied to combinatorial graph problems like the traveling salesman problem [9, 63, 95] and the graph partitioning problem [96]. These problems lend themselves to the use of local search operators because there are a number of very good heuristics for the local improvement of a solution. Other applications include the mapping problem [64] and molecular conformation problems [49]. Mühlenbein [60, 61, 62], Ackley [1], and McInerney [56] have developed application-independent versions of the GA for optimization with local search. In most of these applications, the performance of the GA is substantially improved when the local search technique is employed.

There are a number of common elements to the use of local search in these applications. First, most authors apply the local search to each individual in every generation. A notable exception is work by Mühlenbein et al. [62] who only perform local search if the GA is either not increasing fast enough or if the GA is converging to a solution. Second, most authors apply the local search operator until a local minima was found. Some authors did stop their local search after a fixed time limit or after a fixed number of iterations of their local search algorithm.

Finally, most authors used Lamarckian local search techniques. Belew et al. [7] and Judson et al. [49] make a clear distinction between mutation and local search in their experiments, and were able to compare the performance of Lamarckian and non-Lamarckian local search. They found that Lamarckian local search outperforms non-Lamarckian local search. Judson et al. also found that this performance difference increased as the dimensionality of their problem increased.

In all of these results, the algorithmic design of the GA was not significantly modified to accommodate the local search operator. Some authors did develop parallel algorithms that were more efficient, but they did not introduce any mechanisms that treated local search differently. The one change that most of the authors acknowledged was the use of unusually small population sizes. Few of the experiments performed by the authors used population sizes over 50 and many of them were less than 25. This choice appears to have been made because of computational constraints. In fact, several of the authors noted that their performance improved as the population size was increased.

Chapter III

Local Search with Nonadaptive Global Search

This chapter presents a theoretical analysis of a simple global optimization algorithm that has been modified to selectively apply local search. The algorithm, called probabilistic multistart, is a variant of multistart local search. Instead of applying local search to every randomly generated point, probabilistic multistart applies local search with a fixed probability. This is not a powerful method of selecting samples, since it does not use any information about previously selected samples. But for a given function, the optimal probability of local search is not immediately obvious.

My analysis shows that for any function, the optimal probability of local search is always either zero or one. Note that Monte Carlo corresponds to a probability of zero, while multistart corresponds to a probability of one. Thus, probabilistics multistart is never more efficient than both Monte Carlo sampling or multistart local search. My analysis describes how characteristics of the function, along with the error threshold used for optimization interact to determine whether Monte Carlo or multistart is most efficient.

III.A Definitions

I analyze the computational complexity of the following algorithms:

Monte Carlo sampling (MC) the algorithm that takes n samples from the search space from a fixed distribution.

multistart (MS) the algorithm that takes n samples from the search space from a fixed distribution and applies a complete local search to each of these points.

probabilistic multistart (λ-MS) the algorithm that takes n samples from the search space from a fixed distribution and applies a complete local search from each of these points with probability λ.

These algorithms can be described by a process that iteratively generates a random point and then applies an operator $L(x)$ to the point to generate a final solution point. For MC, the operator is the identity. For MS, the operator is a local search algorithm. For λ-MS, the operator is a combination of the two that applies a local search algorithm with probability λ. Let the operator for λ-MS be $L_\lambda(x)$. Note that the operator for MC is $L_0(x)$ and the operator for MS is $L_1(x)$.

Let $Y = \{y_1, \ldots, y_n\}$ be the set of initial random points used by these algorithms. To compare these algorithms, I assume that they use the same fixed distribution. Without loss of generality, let this be the uniform distribution over some domain D. Let $X^* \subseteq D$ be the set of global optima and let $f^* = f(x^*)$, $x^* \in X^*$. Let $\hat{x}_n$ be the estimate of the global optimum after n samples:

$$\hat{x}_n = \arg\min_{y \in Y} f(L(y))$$

and $\hat{f}_n = f(\hat{x}_n)$. Suppose f has N local minima with domains of attraction D_i such that $D = \cup_{i=1}^{N} D_i$.[1] Let

$$x_i^* = \arg\min_{x \in D_i} f(x)$$

[1]This assumes that saddle points are arbitrarily assigned to one of the basins of attraction.

and $f_i^* = f(x_i^*)$. Finally, let μ be a measure on the Borel sets of $\mathbf{R}^n$. Typically $\mu(A)$ is simply the n-dimensional volume of the set A, more generally μ is a Lebesgue measure.

III.B Complexity Analysis

In computational analysis, the fundamental tradeoff is between computational expense and the performance measure for the problem at hand. In the following analysis, I equate computational expense with the amount of time an algorithm uses. The following complexity analysis considers algorithms that use randomization information [94]. The analysis examines the complexity for the worst possible set of randomization information, except that a δ probability of finding a solution with accuracy greater than ϵ is given.

This complexity analysis concerns the time complexity of the algorithms, and the space complexity of the algorithms is ignored. For this analysis, I assume that every function evaluation incurs a fixed time cost κ. Section IV.C.3 discusses the methods used to evaluate the performance of practical global optimization methods.

III.C Monte Carlo vs. Multistart

I begin by comparing the computational complexities of MC and MS. Figure III.1 illustrates the principal definitions that are used throughout this analysis. The x_i^* are the local optima of this function, and ϵ is the accuracy at which the optimization is to be performed. $A_1(\epsilon)$ and $A_2(\epsilon)$ measure the amount of each local minimum which contains solutions that are ϵ-accuracy (represented by the gray shaded region with the bar underneath). $B_1(\epsilon)$ and $B_2(\epsilon)$ measure the size of each local minimum (represented by the bars connected by dashes to the curve). The proportions of these values to the size of the entire search space are the quantities used in this analysis.

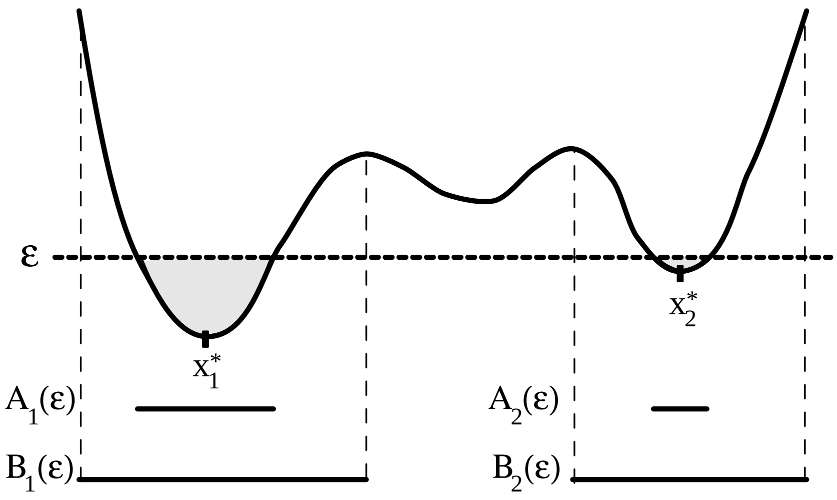


Figure III.1: Illustration of definitions of $A_i(\epsilon)$ and $B_i(\epsilon)$.

Formally, let

$$\begin{aligned} A_i(\epsilon) &= \{x \in D_i \mid Err(x) \leq \epsilon\} \\ \alpha &= \sum_i \frac{\mu(A_i(\epsilon))}{\mu(D)} \end{aligned}$$

and let

$$\begin{aligned} B_i(\epsilon) &= \{D_i \mid Err(x_i^*) \leq \epsilon\} \\ \beta &= \sum_i \frac{\mu(B_i(\epsilon))}{\mu(D)} \end{aligned}$$

α is the fraction of points in the domain D that are ϵ-close, while β is the fraction of points in D that are in basins of attraction that contain points that are ϵ close. Clearly, $\beta \geq \alpha$.

III.C.1 Monte Carlo Complexity

Given n samples, the probability that the solution is ϵ-close is

$$P(Err(\hat{x}_n) \leq \epsilon) = 1 - (1 - \alpha)^n.$$

If we have a δ probability of error, then

$$\begin{aligned} 1-\delta &= 1-(1-\alpha)^n \\ \delta &= (1-\alpha)^n \\ n &= \frac{\log(\delta)}{\log(1-\alpha)} \end{aligned}$$

Since each sample requires a single function evaluation, the complexity of MC is

$$\frac{\kappa \log(\delta)}{\log(1-\alpha)}.$$

III.C.2 Multistart Complexity

Given n samples, the probability that the solution is ϵ-close is

$$P(Err(\hat{x}_n) \leq \epsilon) = 1-(1-\beta)^n$$

If we have a δ probability of error, then

$$\begin{aligned} 1-\delta &= 1-(1-\beta)^n \\ \delta &= (1-\beta)^n \\ n &= \frac{\log(\delta)}{\log(1-\beta)} \end{aligned}$$

Each sample requires the application of the local search method. If T_{ls} is the expected cost of local searches started in $D - B(\epsilon)$, then the expected complexity is

$$\frac{T_{ls} \log(\delta)}{\log(1-\beta)}.$$

Interesting local search methods require the evaluation of points in the search space, so it is reasonable to assume that $T_{ls} > \kappa$.

III.C.3 Comparison

We can evaluate the relative performance of MC and MS by comparing their complexity for given ϵ and δ. This gives us a feeling for the trade-off between global search and local search by defining the conditions for which each of these algorithms is more efficient. It is better to use MS when its complexity is less than that of MC. This is true when

$$\begin{aligned} \frac{\kappa \log(\delta)}{\log(1-\alpha)} &> \frac{T_{ls}\log(\delta)}{\log(1-\beta)} \\ \frac{\log(1-\beta)}{\log(1-\alpha)} &> \frac{T_{ls}}{\kappa} \end{aligned}$$

which is equivalent to having

$$1-\beta < (1-\alpha)^{T_{ls}/\kappa}$$

To understand this inequality, note that

$$0 \leq 1-\beta \leq 1-\alpha \leq 1.$$

As T_{ls}/κ increases, $(1-\alpha)^{T_{ls}/\kappa}$ approaches zero exponentially fast. Therefore, the inequality will be true if $1-\alpha$ is near one and T_{ls}/κ is not too big.

III.D Probabilistic Multistart

III.D.1 Complexity

We now consider the complexity of λ-MS, for which

$$P(Err(\hat{x}_n) \leq \epsilon) = 1 - z^n,$$

where $z = \lambda(1-\beta) + (1-\lambda)(1-\alpha)$. If we have a δ probability of error, then

$$n = \frac{\log(\delta)}{\log(z)}$$

The expected cost for a sample of size n is

$$\begin{aligned} \sum_{i=0}^{n} (iT_{ls} + (n-i)\kappa) \binom{n}{i} \lambda^i (1-\lambda)^{n-i} &= n\lambda T_{ls} + (n - n\lambda)\kappa \\ &= n(\lambda T_{ls} + (1-\lambda)\kappa) \end{aligned}$$

Therefore, the complexity is

$$\frac{\log(\delta)}{\log(z)}(\lambda T_{ls} + (1-\lambda)\kappa) \tag{III.1}$$

III.D.2 Comparison

I now demonstrate that λ-MS is never more efficient than the best of MC and MS. Let $g(\lambda)$ be the complexity in Equation III.1 parameterized by λ.

If λ-MS is more efficient than both MC and MS, then since $g(\lambda)$ is bounded it assumes its minimal value at $\lambda^* \in \langle 0,1\rangle$. Since $g(\lambda)$ is differentiable on $\langle 0,1\rangle$, $g'(\lambda^*) = 0$. Further, $g''(\lambda^*) \geq 0$ since $g(\lambda^*)$ is minimal. To show that λ-MS is not more efficient than both MC and MS, it suffices to show that $\forall\lambda \in \langle 0,1\rangle$ these two conditions do not hold.

Theorem 1 If $T_{ls} > \kappa$, $\beta > \alpha$ and $\delta < 1$, then $\forall\lambda^* \in \langle 0,1\rangle$ s.t. $g'(\lambda^*) = 0$, $g''(\lambda^*) < 0$.

Proof:

Recall that the cost function is

$$g(\lambda) = \frac{\log(\delta)}{\log(z)}\left(\lambda T_{ls} + \kappa - \lambda\kappa\right),$$

which has derivatives

$$\begin{aligned}
g'(\lambda) &= \frac{\log(\delta)}{z\log^2(z)}\left(z(T_{ls}-\kappa)\log(z) - (\lambda T_{ls}+\kappa-\lambda\kappa)(\alpha-\beta)\right)\\
g''(\lambda) &= \frac{\log(\delta)(\alpha-\beta)}{z^2\log^3(z)}\cdot\\
&\quad\left(-2(\alpha-\beta)(\lambda T_{ls}+\kappa-\lambda\kappa) + [(\alpha-\beta)(\lambda T_{ls}+\kappa-\lambda\kappa) + 2z(T_{ls}-\kappa)]\log(z)\right).
\end{aligned}$$

If $g'(\lambda) = 0$, then

$$\log(z) = \frac{(T_{ls}+\kappa-\lambda\kappa)(\alpha-\beta)}{z(T_{ls}-\kappa)}.$$

Substituting into the expression for $g''(\lambda)$, we get

$$\begin{aligned}
g''(\lambda) &= \frac{\log(\delta)(\alpha-\beta)^2(\lambda T_{ls}+\kappa-\lambda\kappa)}{z^2\log^3(z)}\left(-2 + \frac{(\alpha-\beta)(\lambda T_{ls}+\kappa-\lambda\kappa)+2z(T_{ls}-\kappa)}{z(T_{ls}-\kappa)}\right)\\
g''(\lambda) &= \frac{\log(\delta)(\alpha-\beta)^3(\lambda T_{ls}+\kappa-\lambda\kappa)^2}{z^3\log^3(z)(T_{ls}-\kappa)}
\end{aligned}$$

Since $\alpha < \beta$, $(\alpha-\beta)^3 < 0$. $\delta < 1$ implies that $\log(\delta) < 0$. Thus the numerator is positive. The denominator is negative since $1 > z > 0$, $\log(z) < 0$ and $T_{ls} > \kappa$. Therefore, $g''(\lambda) < 0$. ■

Corollary 1 If $T_{ls} > \kappa > 0$, then $\nexists \lambda \in \langle 0, 1\rangle$ s.t. $g(\lambda) < g(0)$ and $g(\lambda) < g(1)$.

<u>**Proof:**</u>

If $\delta < 1$ and $\beta > \alpha$, then the previous argument uses Theorem 1 to prove the result. If $\delta = 1$ then $g(\lambda) \equiv 0$, so $\nexists \lambda \in \langle 0, 1\rangle$ s.t. $g(\lambda) < g(0) = g(1)$. If $\beta = \alpha$, then $g(\lambda) = \log(\delta)\,(\lambda T_{ls} + \kappa - \lambda\kappa)\,/\log(\beta)$. This is a linear function of λ that is minimized at $\lambda = 0$. Thus, $\nexists \lambda \in \langle 0, 1\rangle$ s.t. $g(\lambda) < g(0)$, so the result is proved. ∎

I noted earlier that it is reasonable to assume that $T_{ls} > \kappa$ since interesting local search methods require function evaluations. With this assumption, λ-MS is never more efficient than both MC and MS.

III.E Summary

In summary, I have described conditions for deciding whether MC or MS are more efficient and have proved that that λ-MC is never more efficient than the best of either of these methods. Note that the comparison between MC and MS implicitly depends upon the ϵ-accuracy level required. For large ϵ values, it is likely that a large fraction of the points will be ϵ-accurate and MC will be most efficient. However, when ϵ is small (as is often the case), MS will be most efficient so long as the local search method is not too expensive.

The negative results for λ-MC indicate that selective local search does not necessarily improve performance. This result is particularly strong, since it applies for any given function. It is not clear from our analysis whether this result will generalize to methods of selective local search which use additional information like the value of the objective function or results from previous local searches. This additional information will certainly prove useful in biasing the selection of local searches, but it is unclear whether it will improve the efficiency of optimization.

Chapter IV

Test Problems and Methods

This chapter motivates the global optimization problems used to experimentally analyze the performance of GA-LS hybrids. First, an analysis of the complexity of the optimization problem for the GA is presented. These results emphasize the difficulty of the global optimization problem for an arbitrary function. I conclude that an analysis of GA-hybrids must pay attention to the relationship between the algorithmic parameters of the GA and the function space from which the fitness function is selected.

Next, I motivate the use of test functions whose domain is in $\mathbf{R}^n$ and describe the three test functions used in the experiments. Finally, I motivate the use of GAs with a floating point representation and describe the genetic operators used with the floating point GA.

IV.A Worst-Case Analysis

There have been many attempts to analyze the computational behavior of the GA, with Holland's schema theorem [42] central to much of this analysis. Using it, we can justify how and why certain bit patterns (schemata) will be propagated from one generation to the next. This can be used to analyze the effectiveness of different genetic operators (see for example Syswerda [89]). Related analysis with

Walsh functions has also proven very rewarding. Walsh functions can be used to analyze the effectiveness of genetic operators, as well as analyze the difficulty of the function being optimized [31, 30].

While these analyses provide some understanding of how GAs perform their search, they have not been able to identify the class of functions that GAs efficiently optimize. Any discussion of the computational complexity of the GA must be relative to a specific class of functions. The assumptions that can be made about the class of functions are often critical to establishing interesting complexity bounds.

To illustrate the importance of selecting an appropriate class of functions, I summarize the analysis in Hart and Belew [36] that considers the GA's computational complexity for a very broad class of functions. I assume that the reader is familiar with formal language theory and follow the notational conventions of Hopcroft and Ullman [43]. Recall that P refers to the class of formal languages that can be recognized by a deterministic Turing machine (TM) in polynomial time. Additionally, both NP and RP refer to the classes of formal languages that can be recognized by nondeterministic TMs in polynomial time. The distinction between the two is that for languages in NP there must exist at least one path of computation (sequence of machine states) that leads to an acceptance of the language, whereas for languages in RP at least half of all computation paths must lead to accepting states. It is known that $P \subseteq NP$, $RP \subseteq NP$ and $P \cap RP \neq \phi$, and it is widely believed that the two inclusions are proper.[1] An algorithm is **efficient** if it completes its computation in polynomial time. In other words, a TM M is efficient if the language it accepts is in P.

IV.A.1 Complexity Analysis

Consider $\mathbf{F}$, the class of all deterministic pseudo-boolean functions f such that $f : B^l \rightarrow \mathbf{Z}$, where $B = \{0, 1\}$. We can formalize the problem that the GA

[1] The reader is referred to Gary and Johnson [24] for an excellent discussion of the complexity differences between P and NP, and to Gill [25] for an exposition of probabilistic computation.

attempts to solve as a combinatorial optimization problem DGA-MAX (following the format of Papadimitriou and Steiglitz [72]):

Definition 1 DGA-MAX The Genetic Algorithm combinatorial maximization problem that (1) uses a deterministic fitness function f and (2) assigns the fitness of the maximally fit individual in a population to the fitness of the population itself. An instance of DGA-MAX consists of the following two parts:
1) an integer l defining the combinatorial space B^l
2) an encoding of a TM M_f, which defines a function $f : B^l \rightarrow \mathbf{Z}$ ■

In order to determine the complexity of DGA-MAX, we need to define a version of this problem as a formal language (using the format of Gary and Johnson [24]).

Definition 2 DGA-MAX INSTANCE: a string encoding integers l, and λ, and a TM M_f that computes a function $f : B^l \rightarrow \mathbf{Z}$ in polynomial time.
QUESTION: Does there exist an $x \in B^l$ s.t. $f(x) > \lambda$? ■

The optimization version of DGA-MAX is more powerful than the formal language version of DGA-MAX. Given a TM that solves the optimization version, we can clearly solve the formal language version. However, it is unknown whether the opposite is true (see Papadimitriou and Steiglitz [72] for further details). Thus, the optimization version is at least as difficult as the formal language version of DGA-MAX.

Hart and Belew prove the following.

Proposition 1 DGA-MAX is NP-complete. ■

If $P \neq NP$, as is widely suspected, this result implies that there does not exist an efficient TM that recognizes DGA-MAX.

Corollary 2 The optimization version of DGA-MAX is NP-hard. ■

This result indicates that there probably does not exist an efficient algorithm to solve the optimization version of DGA-MAX. However, this result only applies to deterministic algorithms. Since the GA is nondeterministic, it could be the case that its nondeterminism allows it to efficiently solve either of the versions of DGA-MAX. For example, it is known that there are languages that can be solved more efficiently by probabilistic TMs than by deterministic TMs [25]. The following corollary demonstrates that even though GAs are stochastic, they still require super-polynomial time to solve DGA-MAX unless $RP = NP$.

Corollary 3 If $RP \neq NP$, then DGA-MAX is not in RP. ∎

IV.A.2 Performance Guarantees

These results demonstrated that it is highly unlikely that there exists an efficient algorithm that solves DGA-MAX, whether it be deterministic or nondeterministic. Given this, we consider what other performance guarantees can or cannot be made for DGA-MAX. It is often the case that you can guarantee performance bounds, even for problems that are NP-complete.

Consider the optimization version of DGA-MAX. Let Opt(I) refer to the value of the optimal value for instance I, and let A(I) refer to the value that algorithm A returns for instance I (we assume that A is an efficient algorithm). We are considering a maximization problem, so $Opt(I) \geq A(I)$ for all algorithms A, and we assume that $A(I) \geq 0$ for all algorithms and for all instances.

There are a number of performance guarantees defined in the literature. We consider the the absolute and asymptotic performance ratios to analyze the difficulty of DGA-MAX. We take the following definitions from Gary and Johnson [24]. Let the ratio $R_A(I) = Opt(I)/A(I)$. We define

- Absolute Performance Ratio R_A:

$$R_A = \inf\{r \geq 1 \mid R_A(I) \leq r, \forall I \in \text{DGA-MAX}\}$$

- Asymptotic Performance Ratio R_A^∞:

$$R_A^\infty = \inf\{r \geq 1 \mid \exists N \in \mathbf{Z}^{>0} \text{ s.t. } \forall I \in \text{DGA-MAX},\ Opt(I) \geq N, R_A(I) \leq r\}$$

- Best Achievable Asymptotic Performance Ratio R_{MIN}(DGA-MAX):

$$R_{MIN}(\text{DGA-MAX}) = \inf\{r \geq 1 \mid \textit{there exists a polynomial time algorithm}$$
$$A \textit{ for } \text{DGA-MAX} \textit{ with } R_A^\infty = r\}$$

R_A^∞ indicates whether we can determine a bound on $R_A(I)$ above some value N, while R_A indicates whether we can determine a bound on $R_A(I)$ for values above $N = 0$. R_{MIN}(DGA-MAX) is the smallest value of R_A^∞ over all possible algorithms A. It is this last performance ratio that we analyze. Hart and Belew prove the following:

Proposition 2 If $P \neq NP$, then $R_{MIN}(\text{DGA-MAX}) = \infty$. ■

This result implies that no deterministic algorithm can provide a performance guarantee on $R_A(I)$ s.t. $R_A(I)$ is less than some fixed r. This is true even if we consider only instances that have optima above fixed thresholds. This is a weak performance guarantee, and given this result we can easily demonstrate that other stronger performance results are not possible.

Corollary 4 If $P \neq NP$, then no polynomial time algorithm A can guarantee that

$$Opt(I) - A(I) \leq \delta, \ \ \forall I$$

for a constant $\delta \in \mathbf{R}^{\geq 0}$. ■

Since these proofs are for deterministic algorithms, they are not directly applicable to the GA. The following related proofs show similar results for nondeterministic algorithms.

Proposition 3 If $RP \neq NP$, then $R_{MIN}(\text{DGA-MAX}) = \infty$. ■

Corollary 5 If $RP \neq NP$, then no probabilistic polynomial time algorithm A can guarantee that

$$Opt(I) - A(I) \leq \delta, \ \ \forall I$$

for a constant $\delta \in \mathbf{R}^{\geq 0}$.

■

IV.B Test Problems

The key to the complexity results in the previous section is the fact that the class of functions is very broad. Thus, it is very difficult to efficiently optimize an arbitrary function from this class. The conclusion I draw from these results is that an analysis of the GA must be made relative to a class of functions that represents important practical problems. As was noted earlier, this element is missing from current computational analyses of the GA.

Some experimental analyses have examined the performance of GAs on classes of functions that are motivated by an analysis of the role of the crossover operator. Forrest and Mitchell [23] and Mitchell, Holland and Forrest [58] have examined the performance of the GA on a subclass of Walsh polynomials. These analyses have yet to make definite predictions of the performance of GAs, but have provided much insight into the way the genetic operators perform search.

The functions used in these analyses of the GA have domains in $\{0,1\}^n$. In my experimental analysis, I perform optimization on continuous functions defined on $\mathbf{R}^n$. I claim that it is easier to analyze experimental results when optimizing these functions, particularly when optimizing with local search methods. In discrete spaces, the neighborhood structure used to search the domain space can have a tremendous influence on the performance of local optimization methods. For example, Weinberger [98] provides a formalism for computing something like the Fourier analysis, but over discrete spaces. Analyses like this indicate how discrete problems vary across their domains. Unfortunately, the results of this analysis appear very specific to the topological structure of the discrete space. Thus, results on one topology may be difficult to generalize to problems that have other topologies.

Optimizing functions defined on $\mathbf{R}^n$ also enables us to make comparisons with algorithms developed in the global optimization literature. Most problems in the testbeds used to evaluate GAs and global optimization algorithms are defined on $\mathbf{R}^n$ [1, 21, 31, 91]. Thus, I evaluate GA-LS hybrids on problems for which we can directly compare my results to other global optimization and evolutionary methods.

The experiments in Chapter V and VI perform optimization on three global optimization test functions on $\mathbf{R}^n$. These problems are *essentially unconstrained.* An essentially unconstrained function over a domain D has the following properties: (a) the global optimum is contained in D, and (b) all local minima of f outside of D are greater than the local minima in D.

The global optimization methods described in Chapter II only assume that the function is almost everywhere continuous. However, these test problems are differentiable everywhere. Our experiments will examine the impact of this additional information for methods that use local search methods that use gradient information.

IV.B.1 Griewank

The Griewank function

$$f(x) = \sum_{i=1}^{n} x_i^2/4000 + 1 - \prod_{i=1}^{n} cos(x_i/\sqrt{i})$$

with dimension $n = 10$ is one of the most difficult global optimization test functions [91]. Figure IV.1 shows a one-dimensional slice of this function, which has been smoothed a bit to remove some of the local minima. The Griewank function contains some 500 local minima in $[-600.0, 600.0]^{10}$, which are concentrated around the global optimum at the origin.

IV.B.2 Modified Griewank

A modified Griewank function

$$f_\sigma(x) = \sigma \sum_{i=1}^{n} x_i^2/4000 + 1 - \prod_{i=1}^{n} cos(x_i/\sqrt{i})$$

varies the weight of the quadratic term in the Griewank function. This function is a bumpy quadratic when σ is one and is a product of cosines when σ is zero. This particular class of functions is interesting because it varies the distance between the

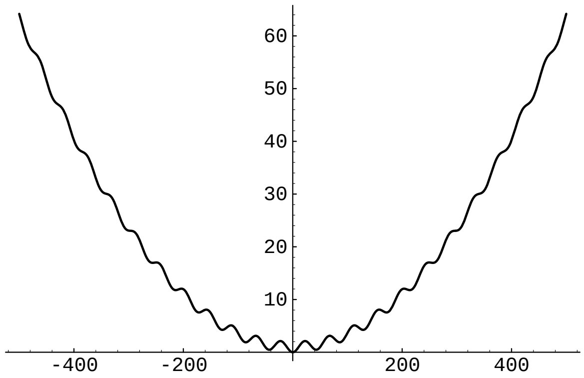


Figure IV.1: The Griewank function.

values of the local minima in the function; as σ approaches zero, the values of the local minima become similar.

I consider this problem in 10 dimensions. For all σ, the minimum value of the function is zero. I optimize this function over the domain $[-600.0, 600.0]^{10}$, and use $\sigma = 0.1$. Figure IV.2 shows a one-dimensional slice of this function.

IV.B.3 Rastrigin

The Rastrigin function

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$$

was proposed in Rastrigin [75]. Mühlenbein, Schomisch and Born [61] describe a modified version of this function

$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) \right),$$

which generalizes Rastrigin's function to an arbitrary number of dimensions. Following Mühlenbein, Schomisch and Born, I optimize this function in 20 dimensions, over the domain $[-5.12, 5.12]^{20}$. The local minima of this function are approximately located on the integer coordinates, and the global minimum is at the origin.

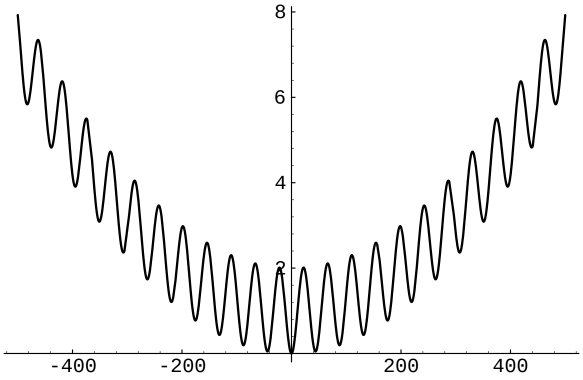


Figure IV.2: The modified Griewank function.

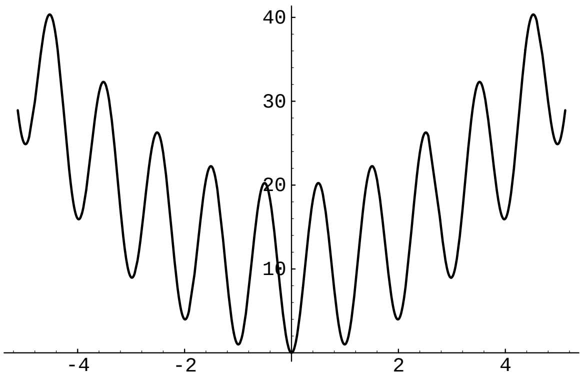


Figure IV.3: The Rastrigin function.

IV.C Optimization Methods

IV.C.1 Floating Point GA

I use a GA with a floating point encoding in the experiments. GAs have traditionally used binary encodings of real numbers to perform optimization on $\mathbf{R}^n$ [16]. While binary encodings have been used to successfully solve optimization problems, special manipulation of this encoding is often necessary to increase the efficiency of the algorithm [83, 100]. There is evidence that optimization on $\mathbf{R}^n$ can and should be performed with real parameters. Goldberg [27] provides formal arguments that floating point GAs manipulate virtual alphabets, a type of schema that is appropriate in $\mathbf{R}^n$. Wright [103] and Janikow and Michalewicz [46] suggests that floating point GAs can be more efficient, provide increase precision, and allow for genetic operators that are more appropriate for a continuous domain.

In the experiments, the panmictic GAs use proportional selection, while the GSGAs use local proportional selection with minimal NEWS neighborhoods. The fitness values are linearly scaled using the average of the worst individual in the last 10 generations. Solutions are scaled between zero and one. Solutions with values below the average are scaled to zero. This type of scaling attempts to make the proportional selection less sensitive to the range of the objective function.

The floating point GA uses a two-point crossover that swaps the floating point values between two individuals. This is analogous to the two-point binary crossover when crossover points are only allowed between the sets of bits that encode the real numbers. A crossover rate of 0.8 was used in the experiments.

IV.C.2 Floating-Point Mutation

I have considered several types of mutation operators for the floating point GA. *Normal mutation* adds a normal deviate to one dimensions of an individual. A random variable Y has a normal distribution denoted by $N(\mu, \sigma)$ if its density

	Rastrigin	Griewank	Modified Griewank
Normal	2.25	23.13	3.55
Cauchy	204.48	0.19	0.48
Interval	15.16	2.02	1.02

Table IV.1: Comparison of floating point mutation operators.

function is

$$f_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-(y-\mu)^2/(2\sigma^2)}$$

The *Cauchy mutation* operator adds a Cauchy deviate to one dimension of an individual. A random variable X has a Cauchy distribution denoted by $C(\alpha, \beta)$ if its density function is

$$f_X(x) = \frac{\beta}{\pi\left\{\beta^2 + (x-\alpha)^2\right\}} \quad \alpha \geq 0, \beta > 0, -\infty < x < \infty$$

The *interval mutation* operator replaces one dimension of an individual with a value uniformly selected over the domain of that dimension.

Table IV.1 shows the performance of GAs using these mutation operators on the three test functions. The results are the best solutions found after 150,000 function evaluations, averaged over 50 trials. The differences between the mutation operators are significantly different.[2]

The Cauchy mutation operator seems a good compromise between the local deviates of the normal mutation operator and the global deviates of the interval mutation operator. The normal mutation operator has a small chance of generating deviates far away, while the interval operator has a small chance of generating nearby solutions. While Cauchy mutation is biased towards small deviates, its distribution has thick tails, which enables it generate very large deviates.

[2]The statistical analyses performed in this dissertation are multiple statistical comparisons between several samples of repeated measurements. In this instance, there are three samples that have 50 measurements each. Multiple comparisons are performed with the GH procedure, which compares samples with unequal variances [90]. This method tests multiple null hypotheses which state that the means of each pair of samples are identical. The confidence of this test applies to all of the hypothesis tests considered collectively. The statistical comparisons reported in this dissertation have a confidence of ($p < 0.05$).

I believe that normal mutation performs better on the Rastrigin function because of the difference in the size of the search domain between the Rastrigin function and the other two functions. The domain of the Rastrigin function is 10.24 wide in every dimension, while the domain of the Griewank and modified Griewank functions are 1200 wide. The width of the search domain can impact the frequency with which the mutation operators generate solutions that are outside the initial search domain (which I call *external* solutions), thereby impacting the ability of the GA to focus its search for the global optimum.

On the Griewank and modified Griewank functions, it is unlikely for the mutation operators to generate external solutions, especially as the search focuses near the origin. On the Rastrigin function, it is not unlikely for the mutation operators to generate external solutions. Comparing the normal and Cauchy mutation operators on this function, it is more likely for Cauchy mutation to generate external solutions since it has a higher probability of generating large deviates. Consequently, the GA using normal mutation is more efficient at minimizing the Rastrigin function.

This suggests that the "width" of the mutation operators be tailored to the size of the search domain. However, such an analysis is beyond the scope of this dissertation. Since I am primarily interested in GA-LS hybrids, the experiments use Cauchy deviates to search the domain with global samples. I use $C(0,1)$ Cauchy deviates with the floating point GA. Unless otherwise specified, the mutation rate is determined by operationalizing the analysis in Schaffer et al. [82] that examines the interaction between population size, mutation rate and the length of the genome. When optimizing in $\mathbf{R}^n$ with a population of size P, the mutation rate used is $\sqrt{e/n}/P$.

IV.C.3 Performance Comparisons

The performance of GA-LS hybrids is compared to Monte Carlo sampling (MC) and multistart local search (MS). In the experiments, the n samples were uniformly selected. The following abbreviations are used for MS and GA-LS hybrids:

MS-SW	- Multistart Solis-Wets
MS-CG	- Multistart conjugate gradient
GA-SW	- Genetic algorithm with Solis-Wets
GA-CG	- Genetic algorithm with conjugate gradient

Previous results suggest that Lamarckian local search is superior to non-Lamarckian local search, so the experiments use Lamarckian local search.

Three different approaches are commonly used to compare the performance of global optimization algorithms. The first is the number of function evaluations needed to find an ϵ-accurate solution. The second is the time needed to find an ϵ-accurate solution. To account for variations in processing speed between computers, the CPU time is normalized by the time needed to evaluate Shekel's function 1000 times [91]. Shekel's function is a standard global optimization test function. The third approach is to compare the performance of the methods after a fixed number of function evaluations. This is particular useful if complete optimization is prohibitively expensive.

This last approach is used in the experiments. The performance measure used to compare optimization algortihms is the value of the best solution found after 150,000 function evaluations. In preliminary experiments, the relative performance of these methods could usually be distinguished after this many function evaluations.

Extra bookkeeping was performed in the GAs to reduce the number of redundant function evaluations. If an individual generated by crossover is identical to one of its parents, or if an individual is selected but not modified by mutation, then the fitness for that individual is not re-evaluated.

When conjugate gradient is used in MS and GA, the gradient evaluation is equated with a single function evaluation. In general, gradient evaluations can be more expensive than function evaluations, but for the test functions the gradient evaluation is well approximated by the cost of the function evaluation. The Rastrigin function's gradient and function evaluations both require $O(n)$ multiplications, additions and calls to trigonometric functions. The Griewank and modified Griewank functions' gradient and function evaluations both require $O(n)$ additions and calls to trigonometric functions. However, the gradient evaluations require $O(n^2)$ multiplications, while the function evaluations require $O(n)$ multiplications. This factor should not substantially bias my results since the dimensionality of these two functions is not large.

Chapter V

Selective Local Search

V.A Introduction

In standard GA-LS hybrids, a complete local search is performed on every individual in the GA's population. While this type of GA-LS hybrid has proven more efficient than the GA on a variety of problems, I propose that a more selective use of local search will improve the efficiency of GA-LS hybrids. This chapter evaluates the efficiency of several GA-LS hybrids that selectively apply local search.

To motivate this work, consider the three graphs in Figure V.1. These graphs represent possible distributions of local minima in an objective function. Figure V.1a is a function for which local search will probably not improve the GA's efficiency. Local search would help refine solutions to the local minima, but the GA's competitive selection should be able to distinguish between points in the two minima since they have distinct ranges. Figure V.1c represents the opposite extreme, where the ranges of the two minima almost overlap completely. I expect that GA-LS hybrids which apply local search frequently will be most efficient on this function, since the fitness of solutions randomly selected from either local minima will not provide reliable information about which local minima contains the global optimum. Finally, Figure V.1b represents an intermediate function for which GA-LS hybrids may be most efficient with a moderate amount of local search.

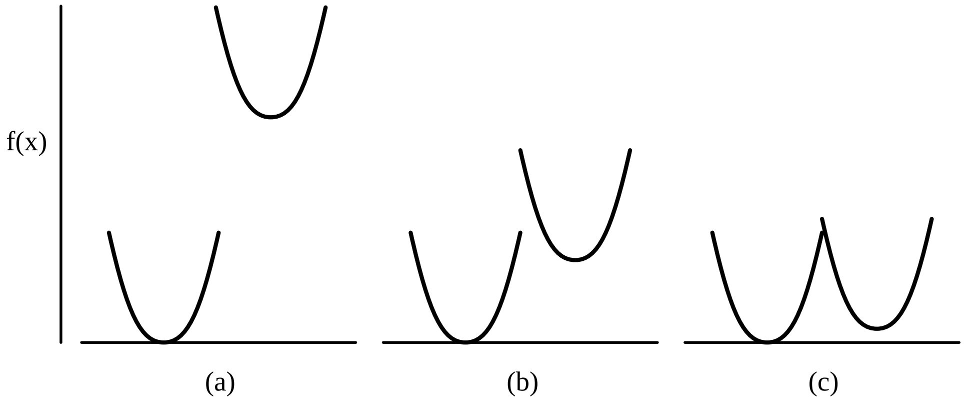


Figure V.1: Three distributions of local minima.

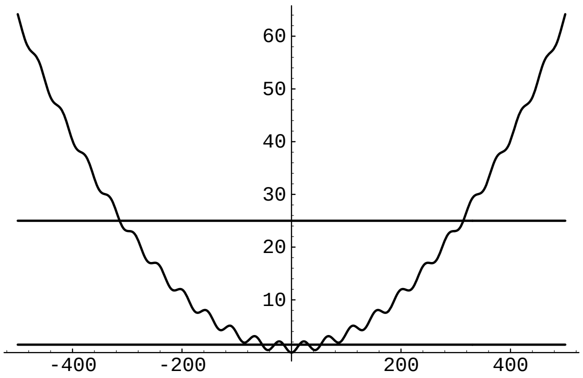


Figure V.2: Two thresholds for the Griewank function.

Recall from Chapter III that the ϵ-accuracy will affect the relative performance of optimization methods. For example, consider Figure V.2 that shows two ϵ thresholds which could be used for the Griewank function. If the ϵ_1 threshold is required, then the objective function is a bumpy quadratic function that has few local minima. However, if the ϵ_2 threshold is used, then the optimization is working with a function with many local minima, all of which have similar local values. This is an important observation because GA-LS hybrids that selectively apply local search may be relatively more efficient at different accuracy levels. The method that best locates the global optimum may be less efficient than other methods when a larger accuracy level is required.

In this chapter, I propose several methods that selectively apply local search. I first describe a non-adaptive method of selecting points with a fixed frequency. This is a simple method, but it provides considerable insight into the role that local search plays in GA-LS hybrids. Next I consider two classes of adaptive methods of selecting local search. *Distribution-based* adaptive methods use redundancy in the population to avoid performing unnecessary local searches. *Fitness-based adaptive* methods use the fitness information in the population to bias the local search towards individuals that have better fitness.

The experiments with these methods address the first two issues described in Chapter I: (I) how often should local search be used and (II) on which solutions should local search be applied? The experiments with fixed frequency local search address issue I, while the experiments with the adaptive methods address issue II. The factors that affect the remaining two issues are addressed as part of these experiments. The interaction between local search length and local search frequency is examined, which addresses issue III. The experiments also compare selective methods for GA-LS hybrids using random local search and conjugate gradient local search. Since conjugate gradient is typically more efficient than random local search, this addresses issue IV.

V.B Nonadaptive Selection

This section examines the performance of GA-LS hybrids that apply local search with a fixed frequency. This type of GA-LS hybrid treats local search like any other genetic operator, and is perhaps the simplest method of selectively applying local search. Much of the previous research with GA-LS hybrids can be viewed as using local search with frequency 1.0.

I expect that reducing the local search frequency will be advantageous when the GA can effectively eliminate regions of the search space in which the global optimum is clearly not located. Reducing the local search frequency lowers the chance of performing unnecessary local searches in these regions. On the other hand, high local search frequencies will be needed when the GA has a difficult time focusing on a single best region and refinement of the samples generated by the genetic operators is needed.

I begin by examining experiments that vary the local search frequency in the GA-LS hybrids. An analysis of these experiments confirms my expectations, and provides insight into the role that local search plays in the GA-LS hybrids. Next, I examine the impact of adding elitism to the GA, and see how it affects the optimal local search frequency. Finally, I examine experiments that vary the population size and local search length.

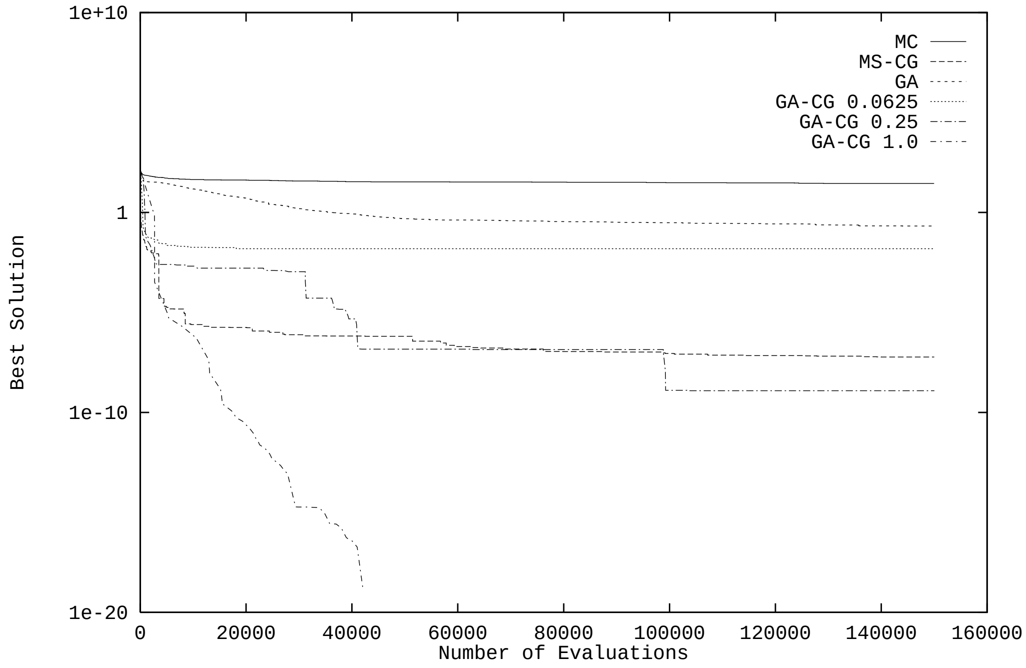
V.B.1 Fixed Frequency Local Search

To measure the effect of the frequency of local search, I examine the performance of GA-LS hybrids that use local search with frequencies 0.0625, 0.25 and 1.0. Experiments were performed with the three test functions. Solis-Wets and conjugate gradient were both run for 50 function evaluations. Populations of size 50 were used. Each experiment was averaged over 20 trials. The optimization algorithms were run until a solution was found whose value was less than 10^{-16} or until 150000 function evaluations were performed. A calculation of the gradient was counted as a single function evaluation (see Section IV.C.3).

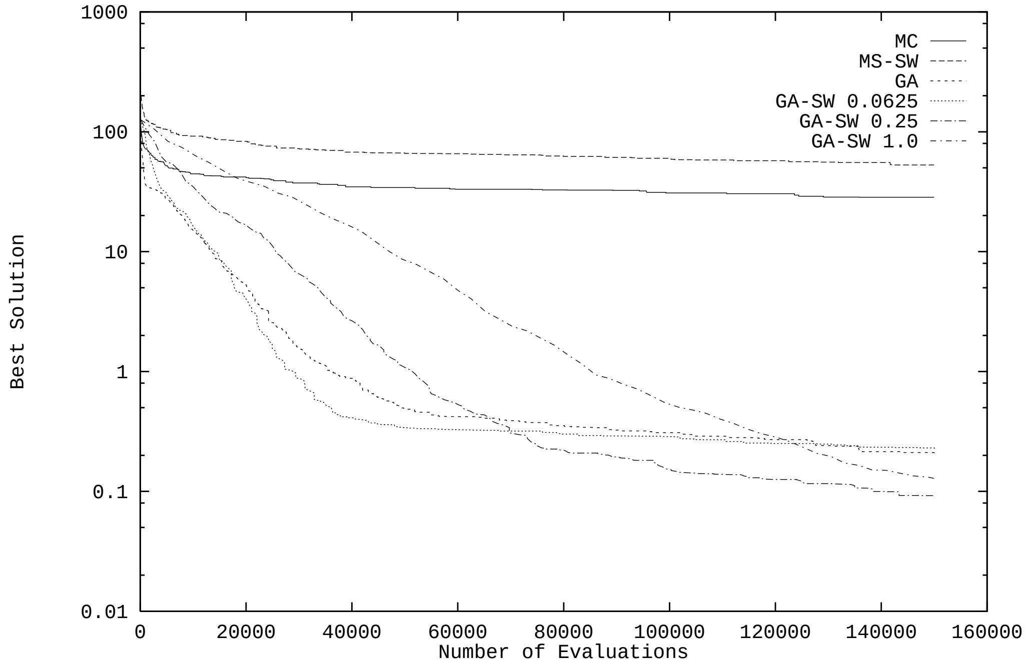
Results

Figures V.3, V.4 and V.5 summarize the results of these experiments.[1] These figures plot the average value of the best solution after a given number of function evaluations. Figures V.3a, V.4a and V.5a compare the performance of MC, MS-CG and the three GA-CG hybrids with different frequencies of local search. Figures V.3b, V.4b and V.5b compare the performance of MC, MS-SW and the three GA-SW hybrids with different frequencies of local search.

[1] The results reported here are an extension of those reported in Hart and Belew [37].

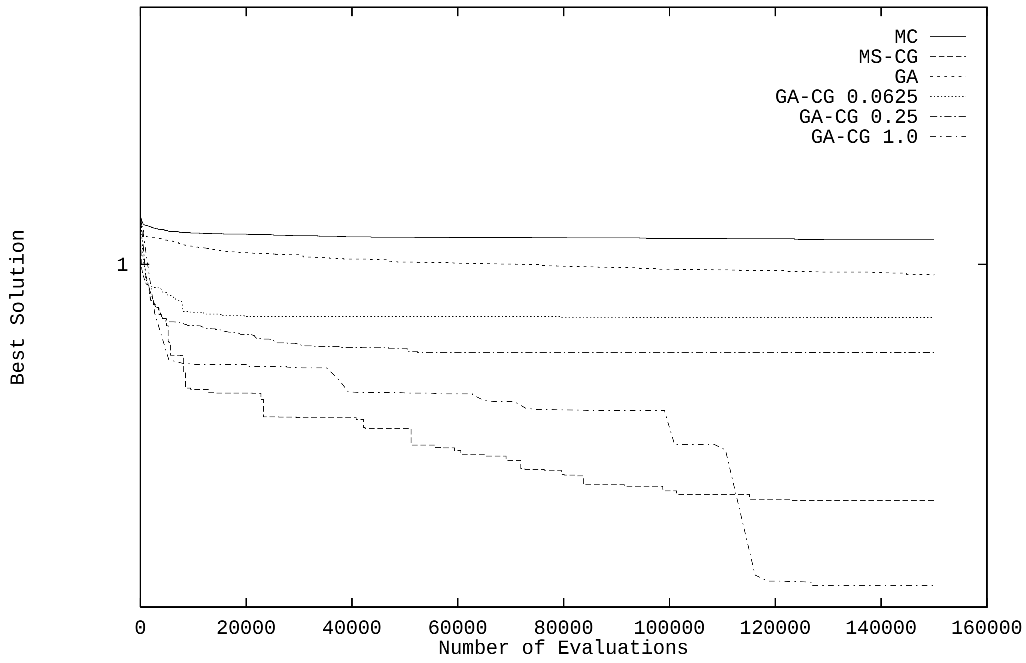


(a)

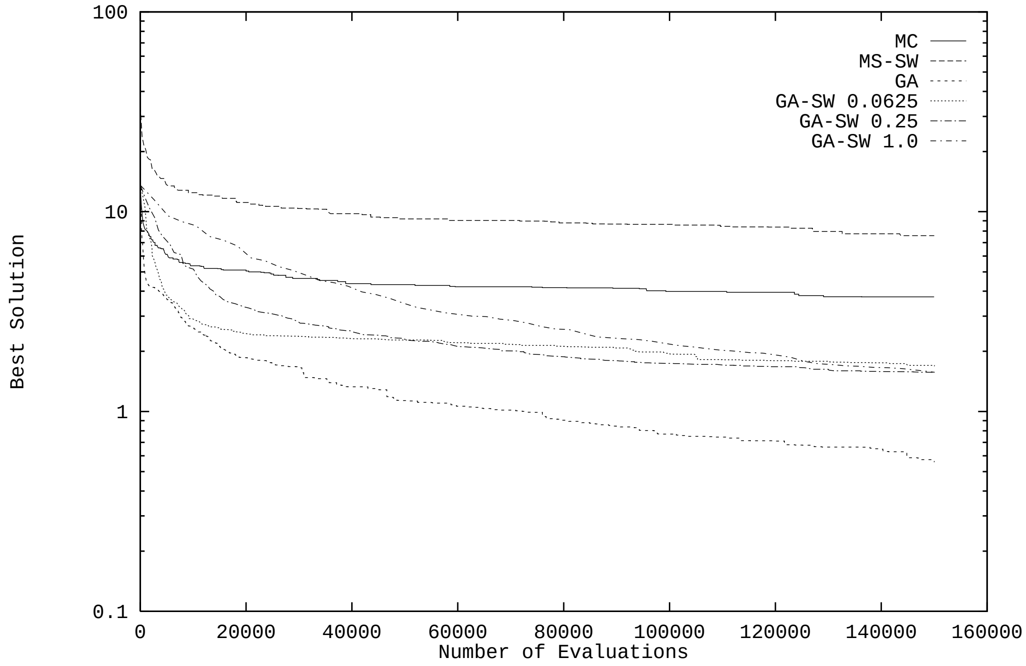


(b)

Figure V.3: Log-performance on the Griewank function using (a) conjugate gradient and (b) Solis-Wets.

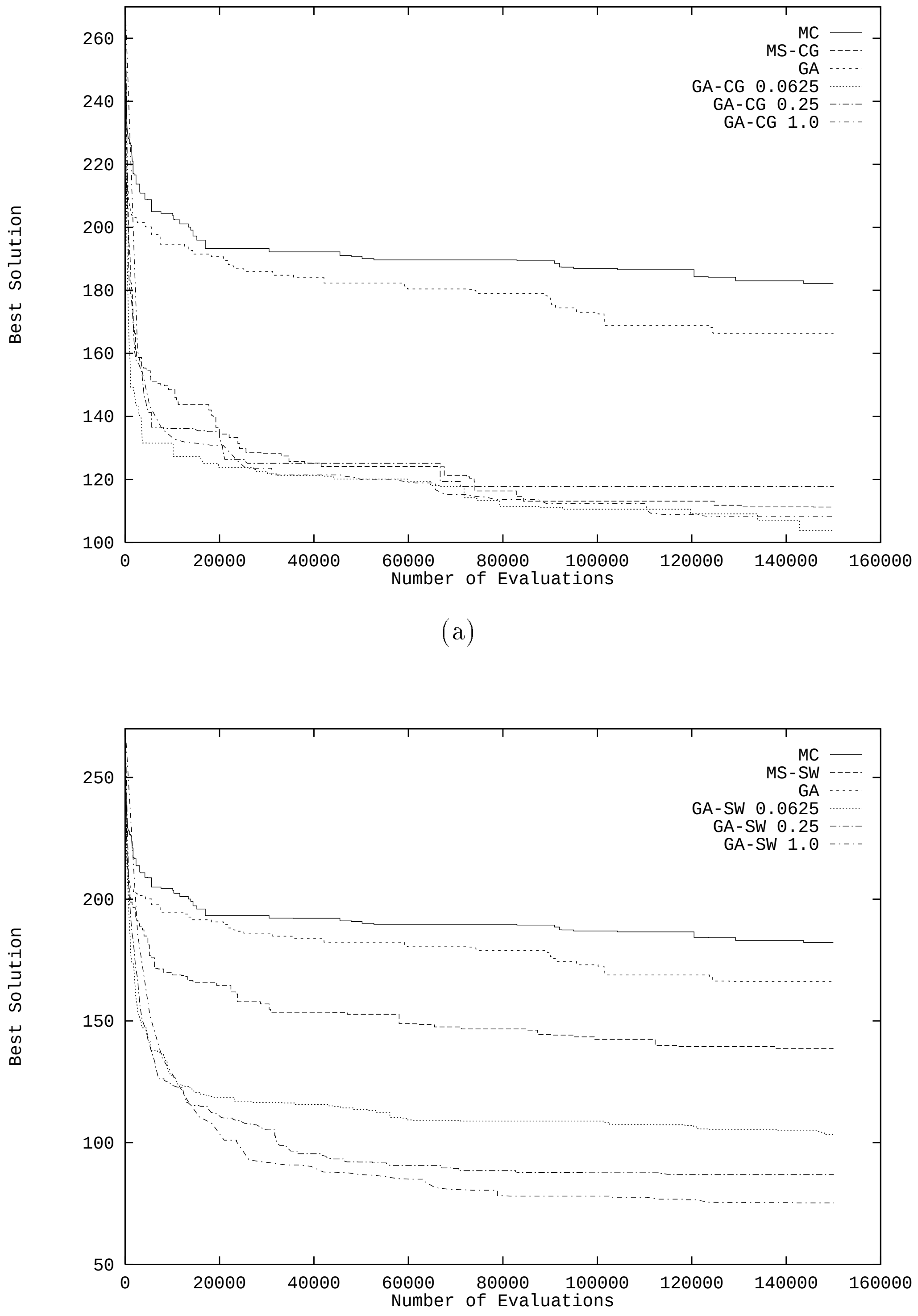


(a)



(b)

Figure V.4: Log-performance on the modified Griewank function using (a) conjugate gradient and (b) Solis-Wets.



(b)

Figure V.5: Performance on the Rastrigin function using (a) conjugate gradient and (b) Solis-Wets.

No single method has the best average performance after 150,000 function evaluations in all six experiments. The GA-LS hybrids using local search with frequency 1.0 have the best average performance for three of the six experiments (see Figures V.3a, V.4a and V.5b). The GA has the best performance when compared with methods using Solis-Wets on the modified Griewank function (see Figure V.4a). In the remaining two experiments, no single method is clearly better.

The performance of MC is relatively poor on all of these functions, though it is better than MS-SW on the Griewank and modified Griewank functions. The performance of MS-CG is quite good for these problems. It is consistently better than the GA, and it is initially better than the GA-CG hybrids on the modified Griewank function.

Discussion

As a group, the relative performance of the GA-LS hybrids was roughly the same for both test functions. Since the objective function has a value of zero at the global minimum of the test functions, we can equate the value of a solution found with the ϵ-accuracy of the solution. An inspection of the results indicates that the GAs which used local search infrequently were more efficient at solving for solutions with relatively large ϵ-accuracy. The GAs that used local search with frequency 1.0 were more efficient at solving for solutions with small ϵ-accuracy, and appear to be best suited for solving for the global optima of these problems. This pattern is most apparent in the performance of the GA-SW hybrids, for which low frequency GA-SW hybrids are relatively more efficient for are wider range of ϵ-accuracies (e.g. see Figure V.4b).

These observations about the relative performance of the GA-LS hybrids provide insight into the way local search is used by the GA. In particular, they indicate that the performance of a GA-LS hybrid is affected by the degree to which the population's fitnesses accurately reflect domain-wide characteristics of the function. The experimental results indicate that the GA-LS hybrids using local search infrequently are more efficient at finding solutions with relatively large ϵ-accuracy. This can be attributed to the fact that the GA's initial populations are quite diverse, so the population's fitnesses accurately reflect the domain-wide characteristics of the objective function. Thus, the competitive selection can reliably identify regions that are likely to contain optimal solutions, and applying local search infrequently avoids numerous local searches on individuals located in bad regions of the search domain.

The experimental results also indicate that GA-LS hybrids using local search frequently are more efficient at finding solutions with small ϵ-accuracy. This can be attributed to the fact that the GA's populations typically lose diversity after several generations, and new individuals generated by the genetic operators become focused on some particular subset of the search space. In this case, the population's fitnesses do not reflect the domain-wide characteristics of the objective function. Consequently, the competitive selection cannot reliably identify regions that are likely to contain optimal solutions. Refining individuals with local search can improve the efficiency of the GA-LS hybrid in two ways. First, the local searches may generate better solutions more efficiently than the GA's competitive selection. Second, the fitnesses of the refined solutions may reflect the domain-wide characteristics of the objective function more accurately, especially when complete local searches are performed.

Note that the population diversity is not the only factor that affects the degree to which a population's fitnesses reflect the domain-wide characteristics of the objective function. As I noted in Section V.A, the overlap in ranges among the local minima of the function affects the degree to which fitness information can be used to discriminate between local minima. This factor certainly affects the reliability of the competitive selection, thereby affecting the optimal local search frequency. Consider the general class of modified Griewank functions that are parametrized by σ. The Griewank function is defined by $\sigma = 1.0$, and the modified Griewank function used in these experiments is defined by $\sigma = 0.1$. Frequent local searches will be more efficient sooner when σ is small, because as σ becomes smaller a larger portion of the search space contains local minima that have values which are very similar to the values of the neighboring local minima. A careful inspection of our experimental results with the Griewank and modified Griewank functions confirms this prediction. Extended simulations with the GA-SW hybrids confirmed this prediction when comparing the performance of GA-SW hybrids with frequencies 0.0625 and 0.25, but not when comparing GA-SW hybrids with frequencies 0.0625 and 1.0.

V.B.2 Elitism

The focus of the previous analysis of GA-LS hybrids concerns the relationship between the GA's competitive selection and the frequency of local search. This analysis indicates that the optimal local search frequency is related to the ability of the GA's competitive selection to reliably identify good solutions. As this becomes more difficult, frequent local searches improve the efficiency of the GA-LS hybrid.

Elitist mechanisms play a similar role in GAs, since they are used to provide an *a priori* bias on the relative value of solutions in the population. Elitist mechanisms identify the best individual(s) in a population and insure that they exist in the next generation. Therefore, these mechanisms induce a strong bias based on the rank of individuals in the population.

It is natural to ask how elitism affects the optimal local search frequency. I expect that introducing elitism will reduce the optimal local search frequency to the extent that the bias induced by the elitist mechanisms aids in the GA's competitive search. If GAs using elitist mechanisms are more efficient than standard GAs, then GA-LS hybrids using local search infrequently may be most efficient. In fact, there may be a substantial reduction in the optimal local search frequency since elitist mechanisms are relatively cheap when compared with local search.

The experiments from the previous section were repeated using an elitist mechanism which preserves the single best individual in the GA's population. Table V.1 shows the final results for the three test functions after 150,000 function evaluations. A statistical analysis of these results indicates that the elitist GA and GA-LS hybrids are significantly better than non-elitist GA and GA-LS hybrids on

Optimization Method	Rastrigin		Griewank		Modified Griewank	
	Normal	Elitist	Normal	Elitist	Normal	Elitist
GA	166.3	**3.6**	0.26	0.07	0.56	0.06
GA-SW 0.0625	102.9	18.4	0.22	0.11	1.68	0.93
GA-SW 0.25	86.8	37.9	0.09	0.04	1.57	1.02
GA-SW 1.0	75.3	50.0	0.13	0.08	1.15	1.15
GA-CG 0.0625	103.8	24.2	0.02	$8.6 \cdot 10^{-4}$	0.06	$1.6 \cdot 10^{-3}$
GA-CG 0.25	117.8	40.8	$1.2 \cdot 10^{-9}$	$1.3 \cdot 10^{-19}$	0.01	$1.6 \cdot 10^{-20}$
GA-CG 1.0	108.1	59.0	$1.2 \cdot 10^{-19}$	$\mathbf{1.1 \cdot 10^{-19}}$	$3.2 \cdot 10^{-8}$	$\mathbf{1.4 \cdot 10^{-20}}$

Table V.1: Average performance of GAs and GA-LS hybrids with and without elitism.

the Rastrigin function. On the Griewank and modified Griewank functions, the only significant differences were between the elitist and non-elitist GAs.

These results confirm our expectations. For all three functions, elitism improves the performance for the GA and GA-LS hybrids. The Rastrigin and modified Griewank functions exhibit marked improvement when elitism is introduced, and the best local search frequency shifts from 1.0 to 0.0625. On these functions, the elitist GAs are more efficient than the GA-SW hybrids that we tested, and the elitist GA is better the GA-CG hybrids on the Rastrigin function. It is possible that a GA-LS hybrid with frequency less than 0.0625 is more efficient than the elitist GAs, but these results suggest that elitism may provide a sufficient bias to preclude the need for local search on these two functions.

This last observation is particularly interesting, though not completely unexpected. The discussion of Figure V.1a in Section V.A indicates that there may be some functions for which local search does not improve the efficiency of the GA. In the absence of prior information about the function, these results recommend the use of elitism in the GA-LS hybrids. Further implications of these results are discussed at the end of this chapter.

V.B.3 Population Size and Local Search Length

Population size and local search length are two more factors which can affect the efficiency of GA-LS hybrids. The population size affects the number of samples that the competitive selection uses to generate new individuals. This factor affects the degree to which the population's fitnesses reflect the domain-wide characteristics of the objective function; more samples enable the population to better reflect features of the objective function. Consequently, we expect that GA-LS hybrids using local search infrequently will be more efficient when using larger populations. The effect of the local search length on GA-LS hybrids is less clear. Longer local searches refine each solution more, but it is unclear how the degree of refinement affects the efficiency of a GA-LS hybrid.

The previous experiments have compare the performance of GA-LS hybrids using populations of size 50 and local searches of 50 function evaluations. These are small populations for the GA, and local searches with this many function evaluations do not often completely minimize solutions. To measure the impact of population size and local search length, the experiments in Section V.B.1 were extended to include populations of size 200 and local search frequencies of length 200. The experiments were not performed for GA-CG hybrids. They performed exceptionally well in the previous experiments, so it would be difficult to interpret results of these new experiments. Tables V.2, V.3 and V.4 summarize the results of these experiments.

The final performance is better with larger populations for the Rastrigin and modified Griewank functions. As expected, GA-LS hybrids with infrequent local search are more efficient when using large populations. However, larger populations appear to be more sensitive to the appropriate selection of the local search frequency (e.g. see Table V.3).

The effect of local search length is not particularly clear. For the Griewank and modified Griewank functions the best results use short local searches, but for the Rastrigin function long local searches are better.

In the results reported in Tables V.2, V.3 and V.4, the total computation

LS Len	LS Freq	Pop Size	
		50	200
50	0.0625	102.86	62.85
	0.25	86.82	63.88
	1.0	75.28	76.44
200	0.0625	67.72	**49.85**
	0.25	76.31	55.49
	1.0	58.66	58.84

Table V.2: Effects of local search length and population size on GA-SW hybrids optimizing the Rastrigin function.

LS Len	LS Freq	Pop Size	
		50	200
50	0.0625	0.226	0.128
	0.25	**0.092**	1.273
	1.0	0.128	10.097
200	0.0625	0.165	0.728
	0.25	0.145	2.689
	1.0	0.370	9.439

Table V.3: Effects of local search length and population size on GA-SW hybrids optimizing the Griewank function.

LS Len	LS Freq	Pop Size	
		50	200
50	0.0625	1.684	**0.469**
	0.25	1.568	0.783
	1.0	1.572	2.457
200	0.0625	2.275	0.695
	0.25	2.052	2.017
	1.0	3.885	5.051

Table V.4: Effects of local search length and population size on GA-SW hybrids optimizing the Modified Griewank function.

LS Len	LS Freq	Pop Size	Rastrigin	Griewank	Modified Griewank
200	0.25	200	55.49	2.689	2.017
200	1.0	50	58.66	0.370	3.885
50	1.0	200	76.44	10.097	2.457

Table V.5: GA-SW hybrids that have a fixed computation of 10,000 function evaluations per iteration.

of each iteration of the GA-LS hybrid may vary with the length of local search and population size. Table V.5 summarizes the combinations of local search length, population size and local search frequency for which the GA-LS hybrids perform the same computation in each iteration (10,000 function evaluations in this case). In the Rastrigin and modified Griewank functions, infrequent local search appears to be the most influential factor.

V.C Adaptive Selection

V.C.1 Distribution-base Adaptation

Methods of distribution-based adaptation modify the local search frequency based on the distribution of individuals in the population. These methods aim to reduce the number of local searches used in each generation when there are redundant solutions in the population. I describe methods based on two notions of redundancy. First, I consider redundancy due to duplicate solutions in the population. These methods avoid performing multiple local searches on the same solution by reducing an solution's local search frequency in proportion to the number of duplicate solutions in the population.

Next, I describe how distance metrics can be used to generalize this notion of redundancy. The goal of this generalization is to avoid performing multiple local searches on solutions that are within the same basin of attraction. Since it is difficult to identify whether two solutions are in the same basin of attraction, this generalization reduces the local search frequency in proportion to the number of similar solutions in the population.

Redundancy from Duplicate Solutions

When duplicate solutions exist in a population, it is possible (and perhaps likely) that multiple local searches will be started from the same solution several times. To avoid performing these redundant local searches, we can modify the local search probability for each individual in the following manner. Consider the i-th individual which has solution x_i, and let N_i be the number of solutions in the population which have the solution x_i. Given the specified local search frequency λ, the modified local search frequency for the i-th individual is λ/N_i. The values N_i can be calculated with $O(N^2)$ pairwise comparisons of the population's individuals. Using this method, the expected number of local searches performed on each distinct solution in the population is one. I call this the *complete* method for calculating redundancy.

The complete method of calculating redundancy can be expensive for practical problems which have large populations and high dimensionality. I propose two approximations to this approach that use information from the crossover operations. Recall that in sexual genetics, all new individuals are the result of mating, so we focus on these events. When crossover is performed, we can determine whether the parents of the new individual are duplicates. This is actually information about whether the redundancy in the previous population, but we can use it to estimate the redundancy of the individual in the current population.

Let $\delta(x, y)$ be the Kronecker function; $\delta(x, y)$ is one if x equals y, and is zero otherwise. If the parents of the i-th individual are p_1 and p_2, then we can modify the local search frequency of the i-th individual as follows

$$\frac{\lambda}{1 + \eta(N-1)\delta(p_1, p_2)}$$

where $\eta \in [0, 1]$. This method approximates the complete method of calculating redundancy by assuming that $\eta(N-1)$ other solutions have the same difference as the parents of the current individual. I call this the *local approximation* to the complete method of calculating redundancy, since it modifies each individual's local search frequency independently.

The *global approximation* to the complete method calculates the number of individuals generated by duplicate parents, and makes a global modification to the local search frequency. If there are N' duplicate pairs of parents during crossover, then the modified local search frequency is

$$\frac{\lambda}{1 - N'/N}$$

The modified local search frequency used with the global approximation results in lower search frequency for both the redundant and nonredundant individuals.

Since the crossover operator can only tell us if two particular solutions are identical, it cannot be used to identify the subsets of redundant solutions. Hence the two approximation methods cannot guarantee that local search will be applied to only one solution from every subset of redundant solutions.

Redundancy from distance metrics

The notion of redundancy in the previous section relies on statistics of the number of individuals that are identical. I now describe a generalization of this method that uses a distance metric over the space of genotypes, and demonstrate that the previous method is a special case. The motivation for this new method is that in addition to avoiding multiple local searches on the same solution, we would also like to avoid performing multiple local searches on solutions that are within the same basin of attraction of a local minima. Since it is difficult to identify whether two solutions are in the same basin of attraction, we reduce the local search frequency based on the degree to which solutions in the population are similar.

This generalization is inspired by the biological notion of inbreeding, which uses a measure called the F statistic to quantify the self-similarity of diploid individuals in a population. Appendix A formalizes the similarity measure used in the biological definition of the F statistic and generalizes the F statistic for similarity measures on an arbitrary space.

To use the generalized F statistic, we select a distance metric for the space of solutions and calculate the following ratio

$$F_{IT} = \frac{H_T - d(X,Y)}{H_T}$$

where H_T is the expected distance between solutions uniformly distributed in the space of solutions, and $d(X,Y)$ is the distance between two solutions X and Y. This ratio is used to adapt the local search frequency in a manner similar to the methods described in the previous section. To measure the *complete* redundancy with F statistics, the i-th individual calculates

$$F_i = \frac{1}{N}\sum_{j=1}^{N}\frac{H_T - d(X_i,X_j)}{H_T}$$

For an arbitrary metric, F_{IT} is bounded above by one, and is bounded below by $H_T - \max_{ij} d(X_i,X_j)$, which may be negative. Thus F_i may be negative. With this in mind, the modified local search frequency for the complete method is

$$\begin{cases} \lambda/(NF_i) & F_i \in [1/N,1] \\ \lambda & \text{otherwise} \end{cases}$$

The local and global approximate methods are similarly defined. If F' is the F statistic of the parents of the i-th individual, then the modified local search frequency for the local approximate method is

$$\begin{cases} \lambda/(1+\eta(N-1)F') & F' \in [0,1] \\ \lambda & \text{otherwise} \end{cases}$$

where $\eta \in [0,1]$. If F'' is the sum of the F statistics of the parents used to perform crossover, then the modified local search frequency for the global approximate method is

$$\begin{cases} \lambda/(1-F''/N) & F'' \in [0,1] \\ \lambda & \text{otherwise} \end{cases}$$

Since the GA focuses its sampling in subsets of the search domain, it is possible for the ratio F_{IT} to be high, even though the population contains few redundant solutions. To account for this effect, we can update the value of H_T based on the extents of the solutions in the current population. This modification increases the amount of local search performed at each iteration, enabling local search to be used at every stage of the search.

Finally, I make two observations. First, redundancy due to duplicate solutions can be captured in this framework using

$$d(X,Y) = (X \neq Y) \tag{V.1}$$

With this distance metric, $H_T = 1$ and the value of F_{IT} is $1 - d(X,Y) = (X == Y)$. Thus F_i measures the average number of individuals that are identical to the i-th individual.

Second, note that the complete method is closely related to the the method of *fitness sharing* proposed by Goldberg and Richardson [29]. Fitness sharing is a method of inducing niche behavior in GAs that enables the GA to converge to a population that is distributed over several local optima. This method modifies the fitness measure of every individual in the population based on the distance between each individual with the rest of the population. The modified fitness measure used with fitness sharing is

$$f_s(x_j) = \frac{f(x_j)}{\sum_{i=1}^{n} s(d(x_i, x_j))}$$

where $d(x,y)$ is a distance metric and

$$s(x) = \begin{cases} \frac{\sigma_{\text{share}} - x}{\sigma_{\text{share}}} & \sigma_{\text{share}} \geq x \\ 0 & \text{else} \end{cases}$$

If $\sigma_{\text{share}} = H_T$, then the normalized F_{IT} is $s(d(x,y))$. Thus the denominator of the modified fitness function can be seen as the sum of *normalized* F statistics between the individual and the rest of the population. This is very similar to the calculation performed when using the complete method to calculate redundancy with F statistics!

V.C.2 Fitness-based Adaptation

Methods of fitness-based adaptation modify the local search frequency of an individual based on the relationship of its fitness to the fitnesses of the rest of the population. The goal of these methods is to use fitness information to bias the selection of solutions. These methods assume that individuals with better fitness are more likely to be in the basins of attraction of good local optima.

Using fitness information, we would like to determine modified local search frequencies $p_i > 0$ such that $\lambda p_i \le 1$ and $\sum_i p_i = N$. This last restriction is not necessary for fitness information to be used. However, I will follow this restriction, since it will allow me to make direct comparisons with the experiments using fixed frequency local search.

The value p_i can be calculated using any of a number of selection strategies that have been proposed for the GA [28]. For example, we can use an elitist method which always performs local search on the individual with the best fitness. To insure that $\sum_i p_i = N$, the frequency of the remaining individuals are reduced. If $N\lambda < 1$, then the frequency of the remaining individuals is zero. Let $k = \arg\min_i f(x_i)$. The modified local search frequency is

$$p_k = \begin{cases} N\lambda & N\lambda \le 1 \\ 1 & \text{otherwise} \end{cases} \tag{V.2}$$

$$p_i = \begin{cases} 0 & N\lambda \le 1 \\ (N\lambda - 1)/(N-1) & \text{otherwise} \end{cases} \qquad i \ne k \tag{V.3}$$

V.C.3 Results

GA-LS hybrids using these mechanisms for adaptively selecting local search were evaluated by optimizing the three test functions. The experiments were run with the same setup as the experiments in Section V.B.1, except that populations of size 200 were used. In preliminary experiments with populations of size 50, these methods induced little or no change in the performance of the GA-LS hybrids.

The local and global approximation distribution-based methods were run using three different configurations: (1) using the inequality distance metric defined in Equation V.1, (2) using the squared L_2 norm defined in Appendix A, and (3) using the squared L_2 norm, with adaptively modified H_T values. The local approximation method was run with $\eta = 1.0$, which makes the assumption that the rest of the population is as inbred as the two parents. The complete method was run for the inequality distance metric, but the other two configurations proved prohibitively expensive so results for these experiments are not available.

The results for the distribution-based methods are summarized in Tables V.6, V.7 and V.8. I omit results for GA-CG hybrids on the Griewank and modified Griewank functions. GA-CG hybrids performed exceptionally well on these functions without the adaptive selective methods, so it would be difficult to interpret these results.

A statistical analysis of the results for GA-SW hybrids on the Rastrigin function (Table V.6a) shows that there are few significant differences between these methods, and no significant improvements over the fixed frequency methods. A statistical analysis of the results for GA-CG hybrids on the Rastrigin function (Table V.6b) found significant differences between the the methods using the L_2 norm and the methods using the inequality metric or fixed frequency local search. GA-CG hybrids using non-adaptive H_T are significantly better when using low frequencies of local search, while GA-CG hybrids using adaptive H_T are significantly better for almost all frequencies of local search.

A statistical analysis of the GA-SW hybrids on the Griewank function (Table V.7) found a number of significant differences. Of particular interest is that the GA-LS hybrids using the inequality methods and fixed frequency local search with frequencies 0.25 and 1.0 are significantly worse than the other methods. This is in contrast to the fact that these methods have the best results when used with low frequency. An analysis of the results for the modified Griewank function (Table V.8) also found these significant differences, but the methods using the L_2 norm without

Method	LS Freq	Baseline	Inequality Metric	Squared L_2 Metric	Adaptive Squared L_2 Metric
Fixed Freq	0.0625	62.84			
	0.25	63.88			
	1.0	76.44			
Complete	0.0625		65.43	NA	NA
	0.25		66.79	NA	NA
	1.0		72.47	NA	NA
Local Approx	0.0625		**55.37**	87.15	84.84
	0.25		64.05	72.46	74.27
	1.0		71.43	68.34	66.04
Global Approx	0.0625		59.27	65.74	75.54
	0.25		66.36	59.69	72.94
	1.0		71.71	65.21	67.02

(a)

Method	LS Freq	Baseline	Inequality Metric	Squared L_2 Metric	Adaptive Squared L_2 Metric
Fixed Freq	0.0625	89.28			
	0.25	106.79			
	1.0	106.18			
Complete	0.0625		94.41	NA	NA
	0.25		108.63	NA	NA
	1.0		105.50	NA	NA
Local Approx	0.0625		91.84	58.98	60.61
	0.25		106.25	65.84	64.33
	1.0		105.80	96.28	**58.37**
Global Approx	0.0625		92.50	68.01	64.29
	0.25		111.31	108.56	67.03
	1.0		104.39	101.48	73.15

(b)

Table V.6: Results for the Rastrigin function using (a) GA-SW hybrids and (b) GA-CG hybrids.

Method	LS Freq	Baseline	Inequality Metric	Squared L_2 Metric	Adaptive Squared L_2 Metric
Fixed Freq	0.0625	**0.128**			
	0.25	1.273			
	1.0	10.10			
Complete	0.0625		0.144	NA	NA
	0.25		1.373	NA	NA
	1.0		8.593	NA	NA
Local Approx	0.0625		0.138	0.482	0.466
	0.25		1.322	0.447	0.274
	1.0		8.817	0.377	0.148
Global Approx	0.0625		0.131	0.570	0.207
	0.25		1.262	0.476	0.157
	1.0		9.362	0.652	2.217

Table V.7: Results for the Griewank function using GA-SW hybrids.

Method	LS Freq	Baseline	Inequality Metric	Squared L_2 Metric	Adaptive Squared L_2 Metric
Fixed Freq	0.0625	0.469			
	0.25	0.783			
	1.0	2.457			
Complete	0.0625		0.631	NA	NA
	0.25		0.854	NA	NA
	1.0		2.482	NA	NA
Local Approx	0.0625		0.534	**0.378**	0.504
	0.25		0.882	0.535	0.594
	1.0		2.371	0.659	0.557
Global Approx	0.0625		0.530	0.399	0.562
	0.25		0.836	0.500	0.480
	1.0		2.523	0.584	0.932

Table V.8: Results for GA-SW hybrids on the modified Griewank function.

adaptive H_T gave the best results.

These statistical analyses indicate that GA-LS hybrids using the L_2 metric can be significantly more efficient than the GA-LS hybrids using either the inequality metric or fixed frequency local search. Figures V.6, V.7 and V.8 compare the fixed frequency local search with the local approximation method using the inequality metric and the L_2 metric with non-adaptive and adaptive H_T on the Griewank function. Figure V.6 shows that the inequality metric has virtually no effect on the performance of the GA-LS hybrid. Inspection of the simulations confirmed that this method made very small modifications to the local search frequency. Figure V.7 shows that the L_2 norm with non-adaptive H_T does much better than the fixed frequency method initially, but it gets stuck and subsequently gets beaten by the fixed frequency methods. Inspection of these simulations revealed that the local search frequency is reduced to a point where very few local searches are performed, which may contribute to its poor final performance. Finally, Figure V.8 shows that the L_2 norm with adaptive H_T also does better initially, and with high frequency remains competitive with the best fixed frequency method. Adapting H_T maintains a higher level of local search, while allowing the local approximation method to select individuals for local search.

These observations are true for the results of the modified Griewank function, but the results GA-LS hybrids on the Rastrigin function are less clear. The results for GA-LS hybrids using the global approximation are similar, though the the global approximation methods do not reduce the local search frequency as much as the local approximation methods. In part, this is due to the use of $\eta = 1.0$ with the local approximation methods, which reduces the local search frequency as much as possible.

Recall that the local selection in GSGAs encourages the formation of demes of very similar individuals. Consequently, I expect to find more redundancy in the populations of GSGAs. To see whether this improves the performance of GA-LS hybrids, the previous results using the inequality metric are replicated for serial GSGAs. The results of these experiments are summarized in Table V.9, along with a compar-

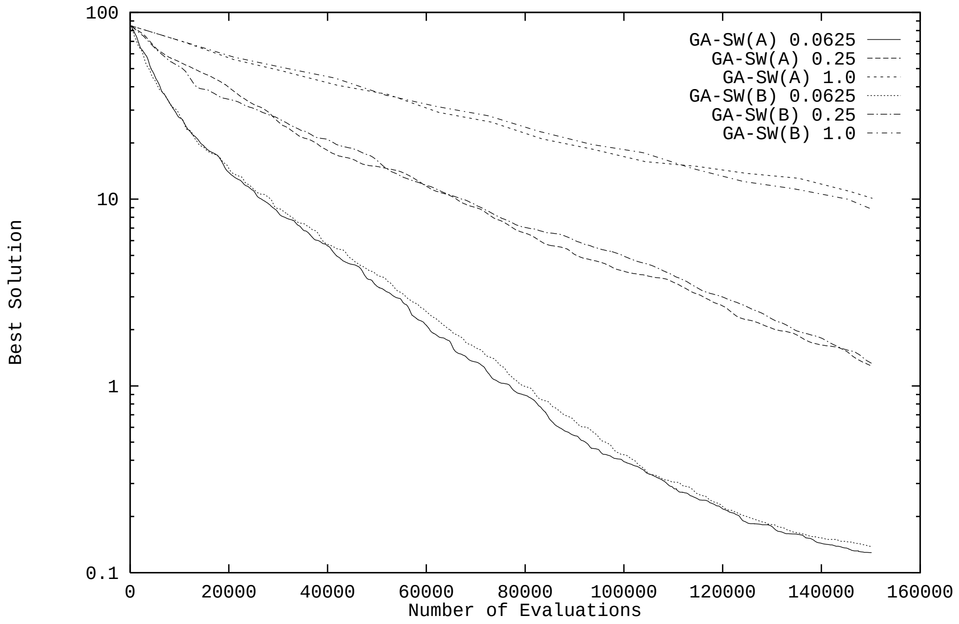


Figure V.6: Comparison of (A) fixed frequency local search and (B) the local approximation method using the inequality metric on the Griewank function.

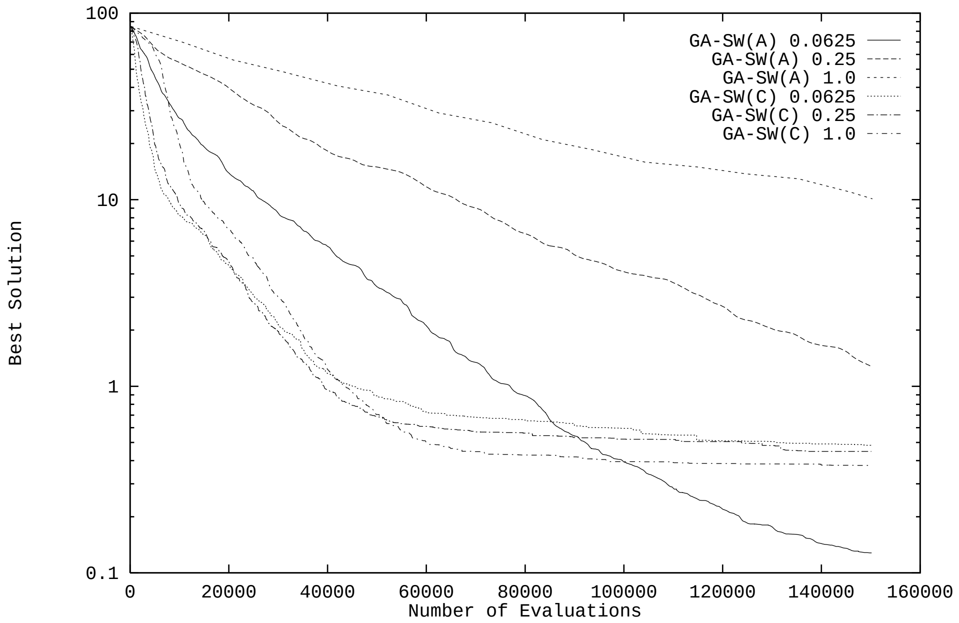


Figure V.7: Comparison of (A) fixed frequency local search and (C) the local approximation method using the L_2 norm with non-adaptive H_T on the Griewank function.

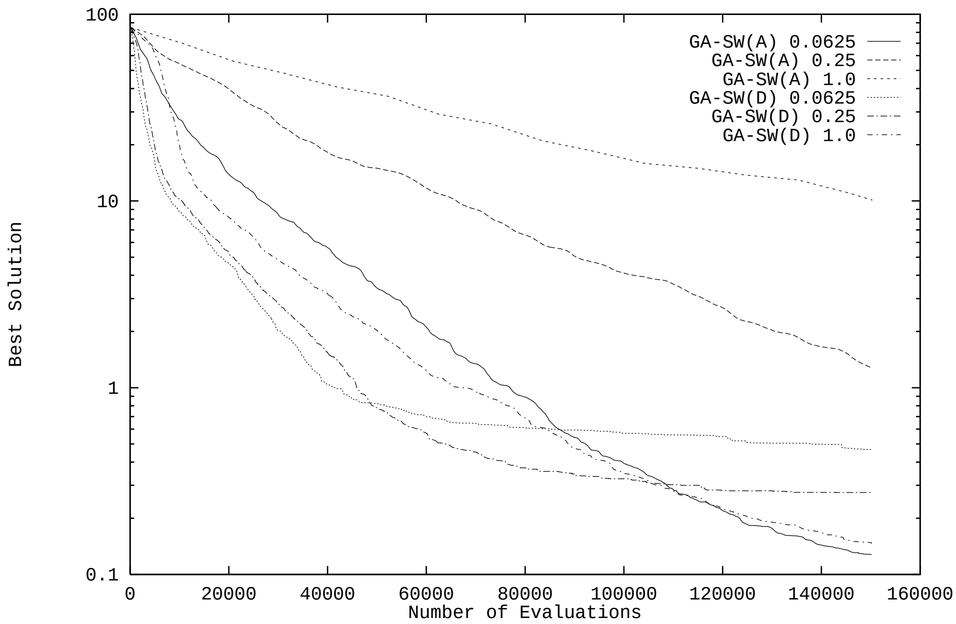


Figure V.8: Comparison of (A) fixed frequency local search and (D) the local approximation method using the L_2 norm with adaptive H_T on the Griewank function.

ison to GA-LS hybrids with fixed frequency local search. For the Rastrigin function, a statistical analysis of these results shows significant differences between the fixed frequency and complete methods, and the local and global approximate methods. No statistical differences are noted for the Griewank and modified Griewank functions.

Finally, I examined the performance of the elitist fitness-based selection of local search. Table V.10 summarizes the results of these experiments. A statistical comparison revealed no statistical differences between these these results and the results for the fixed frequency GA-LS hybrids. A comparison of these results reveals that the results for the Griewank and Modified Griewank functions are virtually identical with both methods. The results for the Rastrigin function appear slightly better with the elitist selection method. This matches the earlier observation that elitism improves the performance of the GA on the Rastrigin function more than the other functions.

Method	LS Freq	Rastrigin	Griewank	Modified Griewank
Fixed Freq	0.0625	8.65	**0.11**	0.458
	0.25	23.94	0.72	0.661
	1.0	61.96	7.40	2.404
Complete	0.0625	8.46	0.134	0.397
	0.25	22.25	0.676	0.5395
	1.0	65.04	7.323	2.538
Local Approx	0.0625	6.65	0.131	**0.326**
	0.25	16.75	0.342	0.446
	1.0	52.62	5.735	1.983
Global Approx	0.0625	**5.16**	0.140	0.442
	0.25	15.03	0.302	0.509
	1.0	57.01	5.878	1.851

Table V.9: Results using GSGAs, comparing fixed frequency local search with methods using the inequality metric.

LS Freq	Rastrigin	Griewank	Modified Griewank
0.0625	56.58	0.12	0.537
0.25	65.50	1.41	0.863
1.0	75.94	10.06	2.532

Table V.10: Results for elitist fitness-based selection of local search.

V.C.4 Discussion

The statistical analyses of these results does not identify a clearly superior method of selecting local search, but the methods using the distance metric with adaptive H_T can be recommended. These methods provided excellent performance on the test functions. Using this type of similarity information is less sensitive to the effects of the local search frequency than the methods using the inequality metric and the elitist method of fitness-based selection. When the distance metric is used without adaptation, it seems to converge prematurely, so adapting H_T seems preferable.

When statistical differences are noted, the complete method of adapting the local search frequency is often significantly worse than the local and global approximate methods. This is unexpected since the complete method preserves the most information about the true state of the population. I believe this happens because the approximate methods tend to lower the local search frequency in the population. This observation suggests that GA-LS hybrids using local search frequencies lower that 0.0625 will be more efficient when using large populations.

One factor which is not explicit in the presentation of these results is that the methods using the L_2 distance metric are computationally more expensive than the methods using the inequality metric. This difference is due to the fact that the L_2 metric is more expensive to compute than the inequality metric. This factor makes a big difference in the methods which measure the complete redundancy, since $O(N^2)$ calls to the distance metric are made every generation. While the results in this chapter have used function evaluations to determine relative time complexities, this observation emphasizes the need to perform time comparisons for methods which may involve substantial overhead.

This factor also makes the GSGAs using the L_2 metric relatively expensive. Since GSGAs have redundant populations, each iteration of the GSGA eventually requires a relatively small number of function evaluations. When this occurs, a considerable fraction of the cost of each of each iteration will be spent performing competitive selection and measuring the redundancy of solutions in the population. The cost of the distance calculations is sufficiently expensive to make the GSGAs using the L_2 metric much more expensive than the GSGAs using the inequality metric.

V.D Summary and Discussion

I have proposed methods of selectively applying local search which apply local search with a fixed frequency, which use information about the redundancy in the population to reduce the frequency of local search, and which apply an *a priori* bias to the selection of individuals for local search. When compared with the standard application of local search to every individual in the population, these methods can offer significant improvements in the efficiency of the search.

The analysis of the results for fixed frequency local search provides a simple model of the interaction between local search and the GA's competitive selection which can be used to design efficient GA-LS hybrids. For example, this model indicates that GAs with large populations will be more efficient when using infrequent local searches. This prediction is confirmed by the experiments in Section V.B.3. These experiments are not comprehensive enough to suggest an optimal balance between population size and local search frequency, but the GA-LS hybrids using large populations with infrequent local search were almost always more efficient than the other combinations. This is an important departure from previous research with GA-LS hybrids, which have typically used GAs with small populations and applied local search to every member of the population.

The model also indicates that GAs that employ biases like elitism will be most efficient with infrequent local search. In fact, the results with the Rastrigin function indicate that elitism may make the competitive selection so powerful that local search is not needed. Since these experiments use a relatively weak form of bias, this raises the question of the utility of local search for more sophisticated GAs that use competitive selection mechanisms which take greater advantage of this type of bias. It is possible that local search may not improve these GAs when applied with the non-adaptive, fixed frequency method. However, we do not expect this to be true for adaptive methods, particularly the fitness-based adaptive methods. These methods employ a bias that should complement these more sophisticated GAs and thereby improve their efficiency.

From these results, it is unclear whether the length of the local searches have a substantial impact on the GA-LS hybrids. However, these experiments confirm that GA-LS hybrids were usually more efficient when more efficient local search methods were employed.

While an analysis of these results does not identify a clearly superior method of adaptively selecting local search, I have argued that the methods using the distance metric with adaptive H_T (i.e. expected distance between solutions in the population) can be recommended. Both the local and global approximation techniques work well with this method. As I have noted above, fitness-based methods are promising, especially in the context of GA-LS hybrids that use more sophisticated GAs.

In all of the methods proposed in this chapter, the initial local search frequency is a parameter that is unspecified. Other factors of the GA like population size and elitist methods may provide a bias for selecting low local search frequency. However, when using adaptive methods of selecting local search points, the best local search frequency is unclear. In part, this is due to the fact that the ϵ-accuracy desired appears to affect the optimal local search frequency. One way to handle this difficulty is to select methods which are relatively insensitive to the local search frequency. The results for the adaptive methods indicate that the methods using the L_2 metric have this property.

Finally, we note that our analysis of GA-LS hybrids may explain the performance that other researchers have observed in their GA-LS hybrids. Davis [59] and Mühlenbein [61] have observed that local search is not needed in the initial stages of the optimization. Our analysis suggests that local search is probably useful for their problems, but is best used with a low frequency.

Chapter VI

Parallel Geographically Structured Genetic Algorithms

VI.A Introduction

Parallel genetic algorithms can be roughly classified according to the type of hardware on which the GA is implemented. *Island-model* genetic algorithms (IMGAs) have been developed for architectures like the nCUBE2 and Intel i860 that exhibit coarse-grained parallelism. These architectures typically have a small number of fairly powerful processors that are loosely coupled. These parallel GAs have been also called coarse-grained GAs. The label "island-model" relates these parallel GAs to models in population genetics that describe the migration of individuals between isolated subpopulations.

Geographically structured genetic algorithms (GSGAs) have been developed for architectures like the CM2 and DAP that exhibit fine-grained parallelism. These architectures typically have on the order of 2^{10} or more simple processors, and are often described as *massively parallel*. These parallel GAs have also been called massively parallel GAs and fine grain GAs. Whitley [101] has identified a subset of these parallel GAs, cellular GAs, which can be equated with finite cellular automaton. The label "geographically structured" refers to the fact that interactions between individuals are structured according to their location on a fixed grid, and bears resemblance to the notion of geographic structure in population genetics.

Another distinction typically seen between the IMGAs and GSGAs concerns the presence or absence of a global synchronization. Massively parallel architectures often use SIMD (Single Instruction Multiple Data) parallelism, which provides global synchronization by executing the same instruction on all of the processors simultaneously. Coarse-grained architectures typically execute instructions in a MIMD (Multiple Instruction Multiple Data) fashion, which does not require (or enforce) global synchronization between the processors.

The algorithms used by IMGAs and GSGAs are clearly distinct from the algorithm employed by the *classic* GA. The classic GA uses a single population of individuals that are panmictically recombined. IMGAs are typically implemented by independently running a classic GA on each processor, with individuals migrated between the subpopulations (see Mühlenbein [63]). GSGAs are implemented by assigning one individual per processor. Selection and recombination is limited to a small number of individuals on neighboring processors, typically forming a two dimensional grid of individuals (see Spiessens and Manderick [87], Collins and Jefferson [12] and McInerney [56]).

Gordon and Whitley [35] have recently argued that the algorithmic nature of these parallel algorithms may be of interest, independent from their implementation on a particular architecture. They experimentally compare the performance of several classic, island-model and geographically structured GAs that are executed on a sequential architecture. They observed that both IMGAs and GSGAs provide performance that is superior to the performance of a classic GA. This philosophy is echoed by Davidor, Yamada and Nakano [14] in their motivation for the ECO framework. The ECO framework provides a serial design for implementing a geographically structured GA.

I am interested in the algorithmic nature of the GSGA, but wish to parallelize it on MIMD architectures. This particular parallel design is motivated by two observations. First, the utilization of a SIMD architecture can be severely reduced if (a) expensive genetic operators are applied to a subset of the population or (b) the cost of the genetic operators is highly variable. In addition, the utilization can be reduced if the genetic operators cannot be synchronized. GSGA-LS hybrids provide an example of both of these cases. GSGA-LS hybrids apply a local search method to individuals in the search space, which searches in a neighborhood of the objective function. The length of a local search may vary depending on the initial point, and local search algorithms like conjugate gradient involve many steps that are not easily synchronized. A MIMD design for a GSGA would not be subject to these penalties, so it would better utilize the parallel architecture.

The second observation concerns the cost of a MIMD design for GSGAs. Several authors have observed that when GSGAs are used, the population forms geographic clusters, or *demes*, containing very similar solutions. Since selection and recombination is performed locally, this implies that a large number of individuals will perform recombination with very similar solutions. In fact, individuals in these demes have a higher probability of performing recombination with an identical solution. If we are optimizing a deterministic fitness function, we can avoid evaluating an individual when this occurs. This can lead to a substantial reduction in the number of function evaluations required by the algorithm. As a result, larger populations can be used than we might have otherwise be expected.

The outline of this chapter is as follows. Section VI.B describes how to map a GSGA onto a collection of processors, and discusses the communication that needs to occur between the processors. Section VI.C analyzes the complexity of the the parallel GSGAs under the assumption that the variability caused by genetic operators can be ignored. Section VI.D extends this analysis to consider the effect of expensive genetic operators like local search. Section VI.E describes the methods used to experimentally confirm our analysis. Section VI.F presents the experimental results, which are discussed in Section VI.G.

VI.B A MIMD GSGA Design

My parallel GSGAs use a toroidal, two-dimensional N_x by N_y grid that is partitioned onto p processors to distribute the computation. Two methods of decomposing the population grid immediately suggest themselves. First, the grid can be partitioned into strips by dividing either the x- or y-dimensions into p parts. Alternatively, the grid can be partitioned into blocks. If r evenly divides N_x and s evenly divides N_y, we can partition the grid onto $p = rs$ processors. Figure VI.1 illustrates these two methods of decomposition. The relative complexities of these decomposition methods will be considered later.

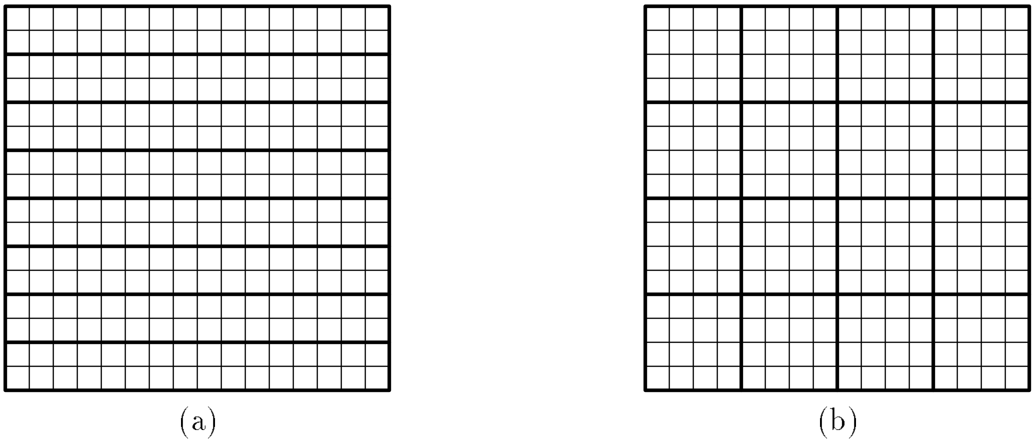


Figure VI.1: Two types of partition methods: (a) strip partitions and (b) box partitions.

To distribute the GSGAs computation, we need to examine the type of communication required between the processors. Communication is required to (1) check for termination signals and (2) perform selection and recombination. Each processor may terminate independently by achieving a specified fitness level or by exceeding a specified number of function evaluations. Communication is required to terminate all of the processors when any of them satisify the termination conditions.

Performing selection and recombination at a given location on the grid requires access to the fitness and genotypes of neighboring individuals that may be located on other processors. Two methods have been used to perform selection and recombination in GSGAs: (1) fixed size neighborhoods have been used to define the set of neighboring individuals, and (2) random walks have been used to stochastically sample the locations of neighboring individuals. My parallel GSGA uses fixed size neighborhoods, so the size of the border areas that need to be communicated between processors can be determined. Figure VI.2 illustrates several fixed-size neighborhoods that could be used with a GSGA. Figure VI.2a and VI.2b are called NEWS neighborhoods, because they only use neighbors to the North, East, West and South.

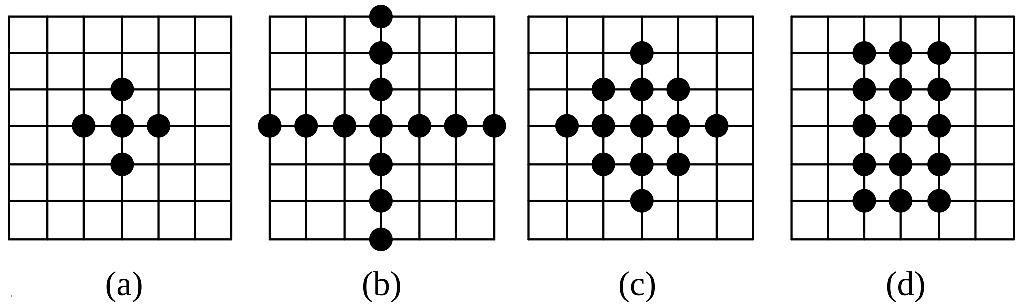
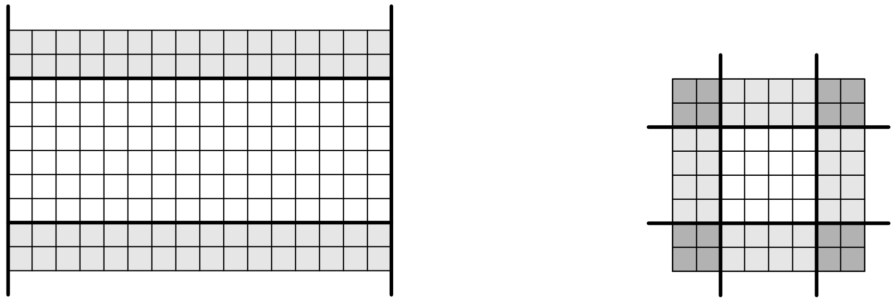


Figure VI.2: Fixed-size neighborhood structures.

Figure VI.3 illustrates the type of border areas needed for the strip and box decomposition methods. Each shaded region represents a border region of individuals that are located on a neighboring processor. Strip decomposition requires two border regions that need to be updated. The number of border regions for the box decomposition method can depend on the neighborhood structure. If a NEWS neighborhood is used with box decomposition, then only the four lightly shaded border areas in Figure VI.3b need to be updated by neighboring processors. If other neighborhoods are used, then the four darkly shaded border areas also need to be updated.

GSGAs can be distinguished by the manner in which interprocess communication is coordinated. Serial GSGAs (A_0) require no communication since all computation is performed on a single processor. Globally synchronized GSGAs (A_1) use global synchronization to guarantee that all border regions for all of the processors



(a) (b)

Figure VI.3: Border areas used by (a) strip partitions and (b) box partitions.

have been communicated before any of the processors can proceed in the next generation. A barrier method is used to globally synchronize the termination check.

Locally synchronized GSGAs (A_2) allow each processor to proceed to the next generation if it has received updated border regions and has satisfied its neighbors' requests for updates to their border regions. Termination signals are also locally synchronized with a small number of short messages.

Asynchronous GSGAs (A_3) do not require each processor to satisfy all requests for updated borders before continuing; only pending requests are satisfied. Further, each processor does not wait to have all of its requests satisfied by its neighbors, but simply updates border regions with the requests that have been satisfied. Asynchronous GSGAs will probably have a faster execution time than the synchronous GSGAs, but processors may frequently be using border regions that are inconsistent with the true state of the parallel system. My experiments examine the impact of this design. Termination signals broadcasted to the processors and are asynchronously checked by each processor.

Finally, independent GSGAs (A_4) do not perform communication to update their border regions. Each processor runs independently, except for asynchronous termination checks. This type of GSGA provides a benchmark for determining the importance of communicating the border regions.

VI.C Complexity Analysis I

Because there are no convergence proofs for the GA, we cannot exactly determine the effect that parallelization will have on the rate at which the GA generates optimal solutions. We can, however, examine the cost of executing k generations. In this section, I present a deterministic complexity analysis. This analysis assumes that we can ignore variability that can occur when the genetic operators are applied, as well as the effects of system load fluctuations. Since the traditional genetic operators are relatively inexpensive, this complexity analysis provides a good characterization of the performance of the GSGA. In the next section, this analysis is extended to consider the variability of the genetic operators.

VI.C.1 Time Complexity

Let T_{gen} be the time to perform selection. Given a neighborhood size s and a problem representation length ρ, the time complexity needed to perform selection and apply the crossover and mutation operators is $O(s + \rho)$ when using local tournament selection, and $O(s \log s + \rho)$ when using local proportional or rank selection [87]. Let T_f is the time to perform a single function evaluation, which is problem dependent, and let T_{flop} is the time needed to perform a single floating point operation. By ignoring variability in the GSGA, we can summarize the work performed for every individual as $(T_{gen} + T_f)T_{flop}$.

To simplify this analysis, let $N_x = N_y = M$ and consider the complexity analysis for a square grid. Let $P = M^2/p$, which is the size of a subpopulation on any processor. Without loss of generality, I assume that p evenly divides into M^2. Let T_{start} be the time needed to initiate a message send, and T_{send} be the time per word need to execute the message send.

The deterministic time complexity for the serial algorithm, A_0, is

$$T_1(k) = O(kM^2T_{flop}\,[T_f + T_{gen}]).$$

The deterministic time complexity for A_i is

$$T_p^i(k) = T_1(k)p^{-1} + kT^i_{comm} \tag{VI.1}$$

where $T^i_{comm} = T_{border} + T^i_{sync}$. The term T_{border} is the cost of sending border information, and the term T^i_{sync} is the synchronization cost.

The synchronization cost, T^i_{sync}, varies for the three parallel algorithms. Algorithm A_1 computes a global termination condition with a log-time spanning tree algorithm, so $T^1_{sync} = O(T_{start}\log_2 p)$. Algorithm A_2 computes a locally synchronized termination condition, by communicating to the N_{nbhr} neighboring processors, so $T^2_{sync} = O(N_{nbhr}T_{start})$. Algorithm A_3 does not compute a synchronized termination condition, so $T^3_{sync} = 0$.

For a given partition of the population grid, T_{border} is the same for all three algorithms. However, T_{border} varies for the different decomposition methods. When using strip decomposition, every processor needs to send two messages to update their neighbors' grids. Suppose every individual on the grid needs to send S words, and suppose there is an overlap of m rows (or columns) between processors. Then

$$T_{border} = 2(T_{start} + mSMT_{send})$$

so

$$T_p^i(k) = T_1(k)p^{-1} + k[2(T_{start} + mSMT_{send}) + T^i_{sync}].$$

When box decomposition is used, the cost of T_{border} depends on the type of neighborhood used by the GSGA. When NEWS neighborhoods are used, every processor needs to send four messages to update their neighbors' grids. If there is an overlap of m rows and columns between processors in the x- and y-dimensions, then

$$T_{border} = 4\left(T_{start} + \frac{mSM}{\sqrt{p}}T_{send}\right)$$

so

$$T_p^i(k) = T_1(k)p^{-1} + k\left[4\left(T_{start} + \frac{mSM}{\sqrt{p}}T_{send}\right) + T^i_{sync}\right].$$

When a more general neighborhood is used, four additional messages are required and

$$T_{border} = 4\left(T_{start} + \left(\frac{mSM}{\sqrt{p}} + m^2S\right)T_{send}\right)$$

so

$$T_p^i(k) = T_1(k)p^{-1} + k\left[4\left(T_{start} + \left(\frac{mSM}{\sqrt{p}} + m^2S\right)T_{send}\right) + T_{sync}^i\right].$$

Note that these time complexities are not exact for the box decomposition when p is small. For example, if there are four processors then the population grid is partitioned into four boxes. If NEWS neighborhoods are used, only two messages are needed since the grid is toroidal and only two processors are neighbors to any given processor.

VI.C.2 Performance Analysis

To measure the performance of the parallel GSGAs, I analyze their *efficiency*. The efficiency of A_i after k iterations is

$$\eta_p^i(k) = \frac{T_1(k)}{pT_p^i(k)}.$$

For the deterministic parallel GSGAs, this expands to

$$\eta_p^i(k) = p^{-1}\left[\frac{1}{p} + \frac{kT_{border}}{T_1(k)} + \frac{kT_{sync}^i}{T_1(k)}\right]^{-1}$$

Now let $\beta_1 = T_{start}/T_{flop}$ and $\beta_2 = T_{send}/T_{flop}$. The efficiencies for the three decompositions are

Strip Decomposition

$$\eta_p^i(k) = p^{-1}\left[\frac{1}{p} + \left(\frac{1}{M^2(T_f + T_{gen})}\right)\left(2\beta_1 + 2mSM\beta_2 + \frac{T_{sync}^i}{T_{flop}}\right)\right]^{-1} \quad \text{(VI.2)}$$

Box Decomposition- NEWS neighborhoods

$$\eta_p^i(k) = p^{-1}\left[\frac{1}{p} + \left(\frac{1}{M^2(T_f + T_{gen})}\right)\left(4\beta_1 + \frac{4mSM\beta_2}{\sqrt{p}} + \frac{T_{sync}^i}{T_{flop}}\right)\right]^{-1} \quad \text{(VI.3)}$$

Box Decomposition- General neighborhoods

$$\eta_p^i(k) = p^{-1}\left[\frac{1}{p} + \left(\frac{1}{M^2(T_f+T_{gen})}\right)\left(8\beta_1 + \frac{4mSM\beta_2}{\sqrt{p}} + 4m^2S\beta_2 + \frac{T_{sync}^i}{T_{flop}}\right)\right]^{-1} \tag{VI.4}$$

All of these efficiencies approach one as M and T_f increase. I expect these two factors will often be large in practice, so the efficiency of the parallel GSGAs should be good.

To analyze the relative utility of the decomposition methods, we compare the efficiencies in equations (VI.2) and (VI.4). Some simple algebra shows that the box decomposition method will have a better efficiency if

$$\frac{\beta_1}{\beta_2} < \frac{mS}{3}\left[M\left(1-\frac{2}{\sqrt{p}}\right) - 2m\right].$$

This inequality indicates that box decomposition becomes more efficient than strip decomposition as p, M and S increase. On many architectures with coarse-grained parallelism, β_1/β_2 is not small; on the nCUBE2, it is approximately 80. Thus strip decomposition may be more efficient when p is small, and if S and M are not too large.

Finally, note that the efficiency of the strip decomposition can be affected by the dimensions of the population grid. If the efficiency for strip decomposition is generalized to an N_x by N_y grid, and partition along the y-dimension, then the efficiency is

$$\eta_p^i(k) = p^{-1}\left[\frac{1}{p} + \left(\frac{2}{(T_f+T_{gen})N_xN_y}\right)(\beta_1 + mSN_x\beta_2) + \frac{T_{sync}^i}{T_{flop}}\right]^{-1}$$

For a fixed population size, this efficiency is maximized when N_x is one. When N_x is one, the population consists of a simple array, and communication is minimal since neighboring processors only need to communicate the neighborhood of a single individual. However, results reported by Gordon and Whitely [35] and others indicate that GSGAs using this type of structured population are less likely to find optimal solutions than GSGAs that use populations structured on a 2D grid.

VI.D Complexity Analysis II

The complexity analysis in the previous section assumes that the application of genetic operators introduces variability that has a minimal effect the complexity of the GSGA. This assumption appears reasonable for GSGAs that employ the standard genetic operators: crossover and mutation. The cost of applying these operators is low relative to the cost of performing the function evaluations and selection. Further, these operators are applied with frequencies near zero and one, which reduces the expected variability that they introduce (see below).

When expensive genetic operators are employed, the variability of the genetic operators must be directly incorporated into the complexity analysis of the GSGA. The following analysis examines the complexity of GSGAs that employ local search. Local search is considered because it is an exemplar of expensive genetic operators, but this analysis applies to any genetic operator.

The deterministic complexity analysis can be extended to include expensive genetic operators by applying a fixed-cost local search to a fixed fraction of the population. An example of a fixed-cost local search is a random local search method, like the one described by Solis and Wets [84], which is terminated after a fixed number of function evaluations. Using local search in this manner, the deterministic computational complexity is a simple extension of our previous analysis. Let λ be the fraction of the population that uses local search, and let T_{ls} be the time complexity of the local search. Then the complexity for A_0 is

$$T_1(k) = O(kT_{flop}\left[M^2(T_f + T_{gen}) + \left\lceil M^2\lambda \right\rceil T_{ls}\right]).$$

The deterministic complexity of A_1, A_2 and A_3 simply uses this value in equation (VI.1).

Indeterminism can be introduced to the GSGA by either applying the local search to a randomly selected subset of the population, or by allowing the cost of the local search to vary. When local search is randomly performed on a subset of the population, the complexity analysis depends on the distribution of the cost of the partitions for each iteration. The cost of a local search can vary if the local search algorithm uses stopping criteria that depend on the characteristics of the function at the current solution. For example, gradient information can be used to terminate local search algorithms when the solution is a critical point of the function. When using variable cost local search, the complexity depends on the sum of the distributions of the cost of the local searches.

VI.D.1 Local Search on Random Subsets

Starting local search on a random subset is simply the fixed frequency local search described in Chapter V. Let λ be the probability that local search is started from each individual. In this case, the complexity of an iteration can be modeled as a binomial random variable Y that assumes values $T_{flop}(T_f + T_{gen})$ and $T_{flop}(T_f + T_{gen} + T_{ls})$, where T_{ls} is the fixed cost of the local search.

Since the cost of each iteration of A_0 is a binomial random variable, the computational cost of A_0 is a sum of k binomially distributed variables with parameters M^2 and λ. Therefore, the computational cost of A_0 can be modeled as a binomially distributed variable with parameters kM^2 and λ. The expected complexity of A_0 is

$$E(T_1(k)) = O\left(kM^2 T_{flop}(T_f + T_{gen} + \lambda T_{ls})\right)$$

Because A_1 is globally synchronized, its complexity is the sum of the complexity of every iteration. The complexity of each iteration is the sum of the cost of communication plus the cost of the longest process. Let Y_i be the cost of the i-th process. The complexity is

$$E(T_p^1(k)) = O\left(kE(Y_{n:n})T_{flop} + kT^1_{comm}\right).$$

where

$$Y_{n:n} = \max_{i=1,\ldots,n} Y_i$$

An analytic expression for $E(Y_{n:n})$ is not available because Y is a binomial random variable. However, when either P or $P/\log p$ is large, we can approximate Y by a normal random variable, Y', with mean μ and standard deviation σ, where

$$\mu = P(T_f + T_{gen} + \lambda T_{ls})$$

and

$$\sigma = T_{ls}\sqrt{\lambda(1-\lambda)P}.$$

Applying the approximation to $Y'_{n:n}$ used in Kruskal and Weiss [53], we have

$$E(T_p^1(k)) \approx O\left(kT_{flop}P(T_f + T_{gen} + \lambda T_{ls}) + kT_{flop}T_{ls}\sqrt{2\lambda(1-\lambda)P\log p} + kT^1_{comm}\right). \tag{VI.5}$$

To analyze A_3, we compare the maximum length of each process independently. Since the computational cost of each process is the sum of k binomial distributed variables with parameters P and λ, the sum is itself a binomial distributed variable, with parameters kP and λ. We can apply the same approximation used for A_1 to get

$$E(T_p^3(k)) \approx O\left(kT_{flop}P(T_f + T_{gen} + \lambda T_{ls}) + T_{flop}T_{ls}\sqrt{2k\lambda(1-\lambda)P\log p} + kT^3_{comm}\right).$$

This is an upper bound on the time complexity, since the communication complexity may be less than T^3_{comm}. It is possible (and perhaps likely) that processors will not receive requests to update all of their neighbors' border regions every generation.

The complexity of A_2 is more difficult to determine, since the cost of the longest processor after k iterations is not independent of the cost of the other processors. It is clear that

$$E(T_p^1(k)) \geq E(T_p^2(k)) \geq E(T_p^3(k)).$$

since synchronization penalizes A_1 more than A_2, and since A_3 is not penalized by synchronization. It is not difficult to show that

$$E(T_p^2(k)) \geq O(E(Y_{2n:2n}) + (k-1)E(Y_{2:2}))$$

but general upper bounds have not been determined.

The efficiency of A_1 and A_3 is

$$E(\eta_p^1(k)) = p^{-1}\left[\frac{1}{p} + \left(\frac{1}{M^2(T_f + T_{gen} + \lambda T_{ls})}\right)\left(B_1 + \frac{T^1_{comm}}{T_{flop}}\right)\right]^{-1}$$

and

$$E(\eta_p^3(k)) = p^{-1}\left[\frac{1}{p} + \left(\frac{1}{M^2(T_f + T_{gen} + \lambda T_{ls})}\right)\left(B_k + \frac{T^3_{comm}}{T_{flop}}\right)\right]^{-1} \tag{VI.6}$$

where

$$B_j = T_{ls}M\sqrt{\frac{2\lambda(1-\lambda)\log p}{jp}} \tag{VI.7}$$

The term B_j represents the penalty introduced by the variability of the local search. This term is zero when λ is zero or one, and is minimized when T_{ls} is small.[1] The term B_j is also minimized when p and j increase. Since B_k is used in equation (VI.6), the efficiency of A_3 improves as the number of iterations of the algorithm increases.

Figure VI.D.1 illustrates how $\eta_p^1(k)$ is affected by λ and T^i_{comm}. Considered as a function of λ, the efficiency has one minima that occurs for small λ. If $T^i_{comm} = 0$, then the efficiency is minimized when B_1 is maximized, which occurs at

$$\lambda = \frac{T_f + T_{gen}}{2(T_f + T_{gen}) + T_{ls}} \leq 0.5$$

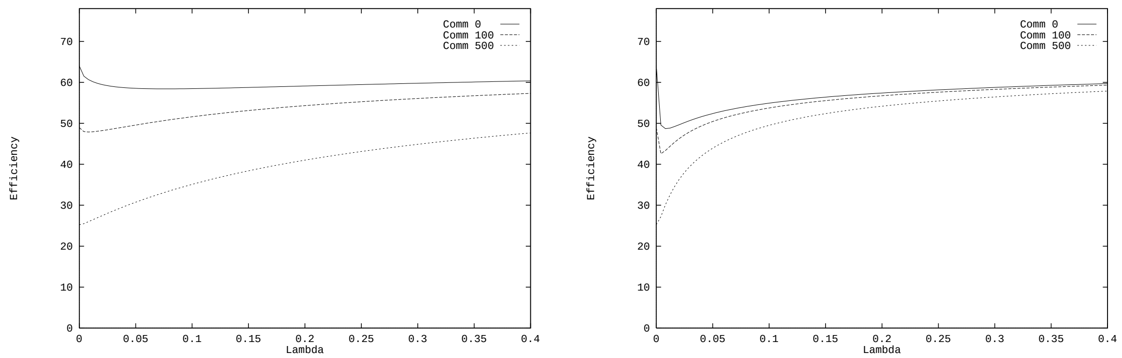
When T_{ls} is much larger than $(T_f + T_{gen})$, the term B_j is maximized when λ is small. As T^i_{comm} increases, the performance for large λ decreases, but the minima moves very little.

VI.D.2 Variable Length Local Search

If the distribution of the cost of the local search is known, the same analysis can be applied to determine the computational complexity. The complexity analysis will be based on the distribution of each processor's iteration, which will be the sum of the distributions of P local searches. Distributions of sums can be analytically determined for many distributions, so this analysis should be straightforward.

I expect that the distribution of the cost of local searches will change as the GSGA samples different regions of the search space. Since the cost of each iteration

[1] This confirms our assumption in section VI.C concerning the minimal impact of the standard genetic operators. Since they are inexpensive and are applied with high or low frequencies, they introduce a small penalty to the overall efficiency.



(a) (b)

Figure VI.4: Comparison of efficiencies for different local search complexities for (a) $T_{ls} = 100T_{flop}$ and (b) $T_{ls} = 1000T_{flop}$. The three curves in the figures contrast the effect of different values of T^i_{comm}. The curves graph $\eta^1_p(k)$ with neighborhoods of size 5, $T_f = 1$, $p = 16$, and $P = 144$.

of A_1 can be independently computed, the complexity analysis in equation (VI.5) can easily be generalized to allow the cost to vary as this distribution changes. However, the complexity of A_3 depends on the distribution of the sum of each processor's costs on each iteration. Thus it may be more difficult to provide a general analysis of the complexity of A_3.

VI.E Methods

To validate the theoretical analysis of the GSGA, I implemented an GSGA and evaluated its performance on the Intel Paragon at the San Diego Supercomputer Center. The GSGA was implemented using the MP++ and LPARX routines described in Kohn and Baden [52]. The LPARX routines were used to implement the inter-process communication in the globally synchronous GSGA, and were modified to perform inter-process communication in the locally synchronous and asynchronous GSGAs.

The parallel GSGAs were evaluated using the Rastrigin function. GSGAs with floating point encoding were used in the experiments. The crossover rate was 0.8

and the mutation rate was 0.01. Proportional selection was used to select individuals within each neighborhood. The theoretical analysis indicates that small neighborhood sizes are most efficient, since this minimizes the total cost of performing selection. In the experiments, I used the *minimal* neighborhood, which only includes the two immediate neighbors along each dimension of the population grid (see Figure VI.2a).

Measuring the efficiency for the parallel GSGAs requires the calculation of both T_p^i, the time to complete for p processors, and T_1^i, the time to complete for one processor. For the parallel GSGAs, the calculation of the efficiency is complicated by the fact that the algorithm for the uniprocessor GSGA differs from the algorithm for the p-processor GSGAin their use of random number generators. On a single processor, one random number generator is used; on p processors, each processor uses a separate random number generator.

To calculate the efficiency, I on an indirect means of measuring T_1^i. Recall that the time complexity for A_i is

$$T_p^i(k) = T_1(k)p^{-1} + kT_{comm}^i$$

This indicates that we can estimate T_1^i by summing the completion times for the p processors and subtracting the time spent performing communication. When analyzing the efficiency in the experiments, the completion times do not include the time required to setup and initialize the GA, but simply include the time required to execute all of the generations.

VI.F Experimental Results

These experiments examine GSGAs run with 4, 16 and 64 processors. Unless otherwise stated, the experiments were run using box decomposition on a square population grid, so the population grid on every processor is square. The GSGAs were terminated when a solution with value 0.1 was found, or after 1000 iterations. All efficiency values are averaged over 10 trials. Number of function evaluations are multiples of 10^4.

GSGA	p	η_p	Num Eval	CPU Seconds
Global	4	0.84	20	104.1
Sync	16	0.81	14	80.5
(A_1)	64	0.75	13	88.0
Local	4	0.85	19	104.9
Sync	16	0.77	14	88.7
(A_2)	64	0.69	13	94.4
	4	0.94	17	79.6
Async	16	0.94	15	71.5
(A_3)	64	0.94	13	64.7
	4	1.00	21	93.9
Indep	16	0.99	17	75.8
(A_4)	64	0.99	15	70.7

Table VI.1: Performance of GSGAs using 24 by 24 grids on each processor.

To match the experimental results to the theoretical analysis, the the efficiency of the GSGAs was calculated. The number of function evaluations reported in the results are the maximum number of function evaluation across the processors. The numer of CPU seconds is the is the maximum number of CPU seconds since the beginning of the first generation across the processors.

VI.F.1 Results without Local Search

Table VI.3 shows the efficiency of the parallel GSGAs when the computation on each processor is kept constant. For these experiments, each processor uses a 24 by 24 grid. When run with 4, 16 and 64 processors, the GSGAs are optimizing with grids of dimension 48, 96 and 192 respectively.

To compare the efficiency of GSGAs for smaller grids, these results are duplicated with each processor using a 12 by 12 grid. When run with 4, 16 and 64 processors, the GSGAs are optimizing with grids of dimension 24, 48 and 96 respectively. Table VI.2 summaries the results of these simulations.[2]

[2]Note that the number of function evaluations differs between A_1 and A_2 because these algorithms employ different termination conditions.

GSGA	p	η_p	Num Eval	CPU Seconds
Global	4	0.67	9.1	61.7
Sync	16	0.57	4.9	48.5
(A_1)	64	0.56	3.5	32.1
Local	4	0.68	9.1	60.6
Sync	16	0.66	5.0	36.3
(A_2)	64	0.62	3.6	30.5
	4	0.85	10.0	57.5
Async	16	0.81	5.4	31.3
(A_3)	64	0.81	3.6	21.0

Table VI.2: Performance of GSGAs using 12 by 12 grids on each processor.

Next, the GSGAs were modified to eliminate redundant function evaluations for individuals that are not modified. Checks were made to insure that crossover did not generate a new individual, and that the individual was not mutated. Table VI.3 summarizes the results of these simulations.

Finally, I compare the performance of the block partition to the strip partition. To compare the strip partition to the results in Table VI.1, every processor uses a 3 by 192 grid. When run with 4, 16 and 64 processors, these GSGAs are using grids of dimension 12 by 192, 48 by 192, and 192 by 192. Table VI.4 summarizes the results of these simulations. Several authors have noted that modifying the dimensions of the population grid may impact the rate at which the algorithm optimizes. Thus, the statistics for the number of function evaluations and iterations may not be directly comparable to the results in Table VI.1.

VI.F.2 Results with Local Search

I examine the impact of local search using 16 processors with a 12 by 12 grid on each processor. Solis-Wets was run for 100 function evaluations, at frequencies 0.0625, 0.25 and 0.5. Table VI.5 summarizes the results of these simulations.

GSGA	p	η_p	Num Eval	CPU Seconds
Global	4	0.84	9.6	92.3
Sync	16	0.82	7.1	68.4
(A_1)	64	0.78	6.8	72.0
Local	4	0.84	9.6	92.7
Sync	16	0.81	7.1	72.3
(A_2)	64	0.74	6.9	81.1
	4	0.94	8.2	69.1
Async	16	0.93	7.6	64.4
(A_3)	64	0.94	7.0	60.3

Table VI.3: Performance of GSGAs using 24 by 24 grids on each processor, with bookkeeping to avoid unnecessary function evaluations.

GSGA	p	η_p	Num Eval	CPU Seconds
Global	4	0.88	16	82.9
Sync	16	0.81	14	78.6
(A_1)	64	0.70	13	89.1
Local	4	0.89	16	80.5
Sync	16	0.83	14	75.7
(A_2)	64	0.71	14	90.2
	4	0.96	18	83.4
Async	16	0.94	15	73.6
(A_3)	64	0.89	13	69.2

Table VI.4: Performance of GSGAs using strip partitioning with 3 by 192 grids on each processor.

GSGA	λ	η_p	Num Eval	CPU Seconds
	0.0	0.57	4.9	48.5
Global	0.0625	0.62	9.3	53.5
Sync	0.25	0.78	22.0	98.3
(A_1)	0.5	0.86	30.0	123.5
	0.0	0.66	5.0	36.3
Local	0.0625	0.64	10.0	54.6
Sync	0.25	0.79	23.0	99.5
(A_2)	0.5	0.86	32.0	125.2
	0.0	0.81	5.4	31.3
Async	0.0625	0.97	9.9	35.9
(A_3)	0.25	0.98	23.0	79.2
	0.5	0.98	32.0	109.8

Table VI.5: Performance of GSGAs using 12 by 12 grids on each processor, with local search frequencies 0.0625, 0.25 and 0.5.

VI.G Discussion

These experimental results are consistent with many of the predictions made by the analysis of the parallel GSGAs. Consider the results in Tables VI.1, VI.2 and VI.3. As expected, the efficiency of the GSGAs is related to the degree of synchronization and communication. GSGA A_1 has the lowest efficiency, since it requires an expensive global synchronization. The efficiency of A_2 is about the same as A_1, but it is a bit higher for smaller grid sizes. The efficiency of A_3 is higher than the synchronized GSGAs, but it is less than A_4, since this algorithm uses no communication between the processors.

Note that the results of the synchronized GSGAs exhibit a decline in efficiency as the number of processors increases. This is consistent with the fact that the synchronization penalty is related to the number of processors that are synchronized. Since A_3 does not use synchronization, the efficiency remains roughly invariant as the number of processors is varied.

The efficiencies in Table VI.1 are higher than those in Table VI.2. This is consistent with the analysis, and reflects the influence of the M parameter. The GSGAs with 24 by 24 grids spend more time working in parallel than the GSGAs using the 12 by 12 grids. Note, however, that the efficiencies in Table VI.3 are comparable to those in Table VI.1. This is surprising since the number of function evaluations reported in Table VI.3 are midway between those reported in Tables VI.1 and VI.2. We might expect to have the efficiencies reduced to a point midway between the efficiencies in Tables VI.1 and VI.2. This suggests that communication costs play a large role in determining the efficiency for our test problem.

The absolute performance of the GSGAs can be compared using the number of function evaluations. Since A_1 and A_2 are synchronized, they have nearly identical performance. They differ only because they use different methods of synchronizing the termination signals, which leads A_2 to use slightly more function evaluations. The performance of A_3 is roughly the same as the synchronized GSGAs, which suggests that it is not adversely affected when border areas are out of date. The performance of A_4 is consistently worse than the other algorithms.

Comparing the efficiencies in Tables VI.1 and VI.4 allows us to contrast the box and strip partitioning. The only factors that differ between the two experiments is that the strip partitioning requires half as many messages as the box partitioning, but communicates four times as many individuals in the population. For A_1, strip partitioning is better than box partitioning for low numbers of processors and is worse for large numbers of processors. This is consistent with our comparison of the analytic efficiency of strip and box partitioning. However, the efficiency of strip partitioning is slightly better for A_2, and is about the same for A_3. This suggests that the method of synchronization may impact the relative utility of the partitioning methods.

Finally, Table VI.5 demonstrates the impact of the local search operator on the efficiency of the GSGAs. As expected, the efficiency increases as the frequency of local search increases, for both the synchronous and asynchronous algorithms.

VI.H Conclusions

The analytic and experimental analysis of MIMD GSGAs demonstrate that they will scale well for large problems. The comparison of the synchronization methods indicates that there is no appreciable penalty (in terms of convergence of the GA) for using asynchronous GSGAs. Since this method has no synchronization penalties, it is more efficient than the synchronous GSGAs. Finally, these results demonstrate that local search can be efficiently used with these parallel GSGAs.

One of the arguments for using SIMD machines is the ability to use very large populations. The largest of these experiments uses 64 processors with 24 by 24 grids. These simulations are executing a parallel GSGA with a 192 by 192 grid that has a total of 36,864 individuals in its population. These GSGAs remain efficient for problems of this size, which indicates that MIMD GSGAs can solve the same problems tackled by SIMD GSGAs. In addition, the MIMD GSGAs offer the ability to use moderate size populations. A comparison of the results in Tables VI.1 and VI.2 shows that using a very large population may be less efficient than using a smaller population, in both CPU time and number of function evaluations.

I expect that these MIMD GSGAs will be competitive with other MIMD GAs, such as the IMGA. Gordon and Whitley [35] show sequential simulations of parallel GAs in which the performance of GSGAs was competitive with other parallel GAs. They note that their simulations used a simple GSGA, and they expect GSGAs to perform very well when more sophisticated methods are employed. One such method is the bookkeeping done by our GSGAs to avoid unnecessary function evaluations. Since GSGAs tend to have greater redundancy in their populations, this may provide MIMD GSGAs an efficiency advantage over other MIMD GAs.

Chapter VII

Applications

To validate the performance of GA-LS hybrids, I have applied them to several practical problems: a neural network problem and to two molecular structural problems. The neural network problem is the six-bit symmetry problem, which has been previously optimized with GA-LS hybrids by Belew, McInerney and Schraudolph [7]. The first molecular structural problem is the problem of solving for a molecule's conformation. This problem has been explored by a number of different authors [49, 55] and there are experiments with GA-LS hybrids for which a comparison is possible. The second molecular structural problem is the problem of docking drug candidates to a target macromolecule [34], which is an important problem in automated drug design. The drug docking results with GA-LS hybrids are compared with simulated annealing, the optimization method used by Goodwell and Olsen [34].

VII.A Neural Networks

Neural networks are simple parametric models that are thought to loosely model biological nervous systems [80]. While there are a variety of types of neural networks, I have examined feedforward neural networks, which perform a deterministic mapping from a set of inputs to a set of outputs [81].

Figure VII.1 illustrates a multilayer feedforward neural network. Each node

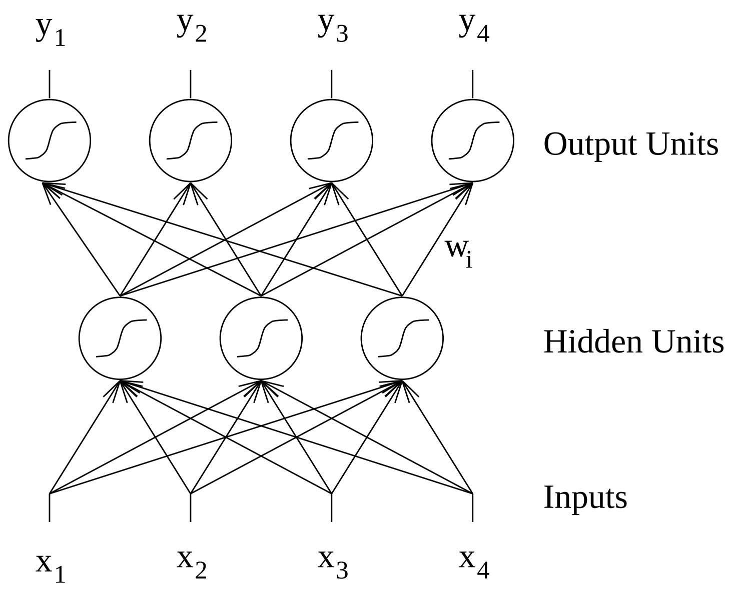


Figure VII.1: Multilayer feedforward neural network with one hidden layer.

in the network computes a weighted sum of the node's inputs that is passed into a logistic function. Let $\{x_1, \ldots, x_n\}$ be n inputs to a node, and $\{w_0, \ldots, w_n\}$ be the $n+1$ weights. Each node computes

$$g\left(w_0 + \sum_{i=1}^{n} w_i f_i\right)$$

where $g(x)$ is the logistic function

$$g(x) = 1/(1 + \exp^{-x}).$$

Because the nodes of the neural network perform a nonlinear transformation of their inputs, feedforward neural networks are capable of performing nonlinear transformations of the inputs.

Suppose we are given a set of data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ for which we would like to know the relationship between the x_i and the y_i. Feedforward neural networks are parametric models of the form $y = f(x, w)$, where w is a vector of the weights (parameters) of the neural network. To estimate the relationship between the x_i and the y_i, minimization techniques are used to determine the weight vector w^* that minimizes

$$J(w) = \sum_{i=1}^{n} E(y_i, f(x_i, w))$$

where $E(\cdot)$ computes the error between the predicted y value and the actual y value, y_i. A common error function is the squared error

$$E(a,b) = \|a-b\|^2$$

When using a smooth error function like this, the gradient of $J(w)$ is computed for feedforward neural networks by *back-propagating* the errors on the outputs through every layer of the network [81].

The weight vector for neural networks is typically large. Therefore, solving neural network problems involves the minimization of an error criterion over a high dimensional search space that has a large number of local minima. In theory, the global minimum of the search space is desired. In practice, minimization is usually performed using local search techniques that can only guarantee solutions which are locally optimal.

Belew, McInerney and Schraudolph [7] use GA-LS hybrids to minimize $J(w)$ for the six-bit symmetry problem. In the six-bit symmetry problem, patterns are classified as one if the left three bits are mirror images of the right three bits. Thus 110011 is classified as one, while 101110 is classified as zero. Their experiments use the GA to control repeated restarts of neural network local searches using back-propagation (BP). Their results indicate that GA-BP hybrids outperform both the GA and multistart local search using BP.

The experiments in this section extend these results in two ways. First, the GA-LS hybrids use local search with various frequencies. Second, the performance of methods using BP is compared with methods using conjugate gradient and *batch* BP. Unlike standard BP, batch BP uses complete calculation of $J(w)$ is used to update the current weight vector. Thus, Equation II.1 becomes

$$\begin{aligned} w_{t+1} &= w_t + \Delta w_t \\ \Delta w_t &= -\eta_t \nabla_w J(w) \end{aligned}$$

Batch BP can be viewed as a simple gradient descent procedure. Unlike BP, the gradient calculation in BP is a reliable estimate of the current descent direction. Hertz, Krogh and Palmer [39] note that the relative performance of BP and batch BP is problem dependent, though BP seems superior in many cases.

Table VII.1 compares the performance of MC, MS, GA and GA-LS hybrids for the three local search methods. The GA-LS hybrids are compared for three fixed frequencies. This is a relatively easy problem, so the more sophisticated methods of applying local search selectively were not performed.

Following Belew, McInerney and Schraudolph, BP and batch BP were run for 200 epochs. To make BP comparable to the other methods, evaluations for an entire epoch are counted as a single function evaluation. These experiments use a three-layer feedforward network with six input units, six hidden units and one output unit. Initial weights for the networks were chosen in the interval $[-0.5, 0.5]$. The learning rule in Equation II.3 is used to perform BP, with $\alpha = 0.5$ and $\eta_t = 2.5/\sqrt{t}$. Belew, McInerney and Schraudolph used $\eta_t = 2.5$, but this form of η_t worked very poorly for batch BP. The large α and η_t values identified by Belew, McInerney and Schraudolph were selected for BP. It is not clear that these are also the best values for batch BP.

Method	Freq	BP	Batch BP	CG
MC		0.0563		
MS		0.0467	0.0546	$1.59 \cdot 10^{-6}$
GA	0.0	0.0341		
	0.0625	0.0018	0.0535	$7.32 \cdot 10^{-7}$
	0.25	**0.0013**	0.0538	$1.51 \cdot 10^{-8}$
	1.0	0.0015	0.0543	$3.30 \cdot 10^{-9}$

Table VII.1: Results for 6-Bit Symmetry.

A statistical analysis of these results shows significant differences between almost every method. In particular, the GA-CG hybrids are significantly better than the GA-BP hybrids, which are significantly better than the GA-Batch-BP hybrids. The GA-LS hybrids appear to be more efficient when used with high frequency of local search, but the statistical analysis does not indicate significant differences between the frequencies of the GA-LS hybrids. It is interesting to note that the methods using batch BP were worse than the GA alone, while the methods using conjugate gradient were better than the GA. This attests the inefficient use the gradient information in batch BP.

VII.B Molecular Conformation

VII.B.1 Introduction

The goal of molecular conformation problems is to solve for the three dimensional structure of a molecule (its tertiary structure), given only a description of the atoms and bonds that comprise the molecule (its primary structure). One approach to solving these problems uses a model of the potential energy of the molecule's conformations that is minimized to find conformations with low energy. Most simple models assume that the conformational energy, V, can be approximated by a sum of different types of energy contributions. For example, we could define V as

$$V = V_{bond} + V_{angle} + V_{torsion} + V_{non-bond} + V_{electrostatic}$$

A detailed description of these terms is given in Le Grand and Merz [54]. Clark, Cramer and Van Opdenbosch [10] describe many of the "standard" force fields. Intramolecular forces are modeled by the terms for bond stretching, bond torsion and angle valence. The bond stretching term is usually represented as

$$V_{bond} = K_{bond}(r_{ij} - r_0)^2$$

where r_{ij} is the distance between the i-th and j-th atoms. V_{bond} is usually defined with a relatively large bond constant, K_{bond}, to hold the bond distance fairly constant at r_0. The bond torsion (dihedral) term measures the energy related to the stresses put on double bonds. This energy is often quite specific to the type of bond that is modeled. Le Grand and Merz [54] distinguish the angle valence energy from other torsion energies

$$V_{angle} = \frac{1}{2}K_{\theta_{ijk}}(\theta_{ijk} - \theta_0)^2$$

where θ_{ijk} is the angle between the bond linking the i-th and j-th atoms and the bond linking the j-th and k-th atoms. Like the bond stretching term, V_{angle} measures the energy added by moving a bond pair from its ideal angle. The intermolecular forces are modeled by the terms for non-bonded interactions and electrostatic interactions. The non-bonded interactions account for van der Walls forces, which are often modeled with Lennard-Jones 12-6 potentials

$$V_{non-bond} = \epsilon\left(\frac{r^*}{r_{ij}}\right)^{12} - 2\epsilon\left(\frac{r^*}{r_{ij}}\right)^6$$

The electrostatic interactions account for interactions between particle charges on atoms. The typical, point charge interaction uses a simple Coulumb expression

$$V_{electrostatic} = \frac{q_i q_j}{\epsilon} r_{ij}$$

where q_i and q_j are particle charges, and ϵ is the dielectric constant of the medium in which the molecule is located.

I consider a simple two dimensional conformation problem that is examined in Judson et al. [49]. This conformation problem concerns a molecule composed of a chain of identical atoms that are connected with rigid rods of length one. The potential energy of this molecule can be modeled by

$$\begin{aligned} V &= V_{non-bond} && \text{(VII.1)} \\ &= \sum_{i=1}^{n-1}\sum_{j=2}^{n}\left[\left(\frac{1}{r_{ij}}\right)^{12} - 2\left(\frac{1}{r_{ij}}\right)^6\right] && \text{(VII.2)} \end{aligned}$$

This equation accounts for the van der Walls forces in the non-bonded interactions.

This function has two types of local minima: knotted and unknotted. Examples of these two types of configurations are shown in Figure VII.2. Figure VII.2a is a knotted configuration, in which the bonds of the molecule cross at some point. Local search methods cannot pull an atom through a knot because there is a very

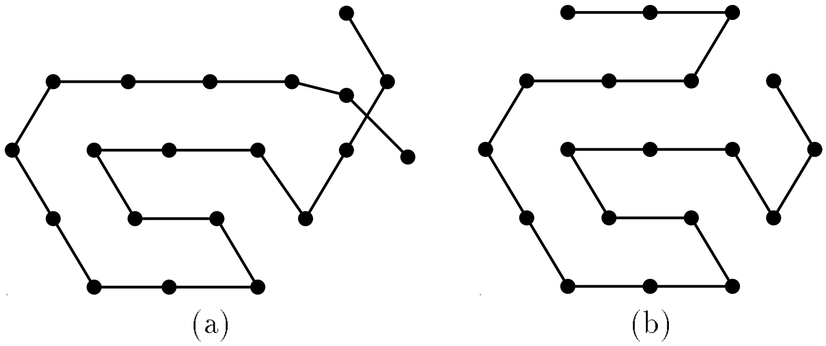


Figure VII.2: Examples of (a) knotted and (b) unknotted configurations. Configuration (b) is a global minimum of the energy function for the 19-atom molecule.

high energy barrier preventing this. The global minima of this function are approximately located on a hexagonal grid with unit spacing. Figure VII.2b is an example of a global minimum. When minimizing a 19-atom molecule, the global minima are known to have a value of -45.3 [49].

VII.B.2 Parametrization

The distance terms r_{ij} can be parameterized in two ways: (1) using the coordinates of the atoms, and (2) using the bond angles and bond lengths. Figure VII.3 illustrates the relation of these parameters to the structure of a simple molecule. Analytic gradients can be calculated for either of these parametrizations, but the angle parametrization is easier to use with GAs since the constraints on the bond lengths are implicit in this representation. Minimizing the energy with the coordinate representation would require the use of constrained optimization techniques to keep the bond lengths at a fixed value.

Since the global minima are approximately located on a hexagonal grid, the angles of the global minima are near multiples of $\pi/3$. We can use this information about the problem to discretize the space of angles, thereby reducing the space of solutions that are searched by the GA. However, the solution must still be minimized in the continuous space of angles, since the optimal solution may not be an exact

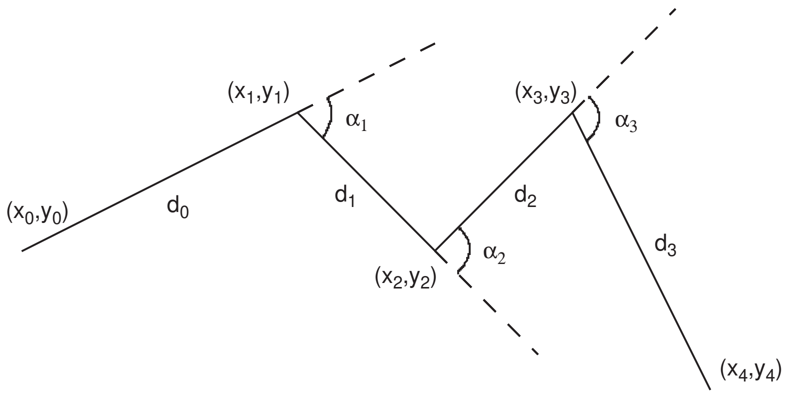


Figure VII.3: Illustration of a simple molecule with dimensions used to parametrized the potential energy.

multiple of $\pi/3$. Thus the GA and local search routines are searching different spaces.

This is an example of a *genotype-phenotype distinction*, which was mentioned in Section II.C.3. A maturation function is used to map the genotype generated by the GA into the space of phenotypes used by the local search method. In this case, the mapping is simply the identity map. To perform Lamarckian local search, a reverse map is also needed. In the experiments reported below, the reverse map simply discretizes the angle θ to θ' using the following rule:

$$\theta' = \left\lfloor \frac{3\theta + \pi/2}{\pi} \right\rfloor$$

VII.B.3 Results

I have tested the efficiency of GA-LS hybrids using both the angle and discrete angle representations. For comparison, the performance of MC, MS and the GA have also been tested. A GA with an integer representation is used to search the discretized angle space. The mutation operator for this integer GA changes a single integer to a uniformly selected integer in the set of possible discretized angles. The MC and MS methods were also modified to uniformly generate discrete angles and then perform local search in the continuous angle space.

Method	Angles	Discr Angles
MC	-27.76	-26.83
MS-SW	**-30.74**	-30.45
MS-CG	-29.09	-28.67
GA	-28.57	**-31.27**

Table VII.2: Results for MC, MS and GA on the 2D conformation problem.

Method	LS Freq	Angles		Discrete Angles	
		GA-SW	GA-CG	GA-SW	GA-CG
Fixed	0.0625	-32.47	-30.93	-42.23	-40.88
Freq	0.25	-32.86	-31.30	-42.12	-38.21
	1.0	-33.71	-31.16	-40.56	-36.51
Inequ	0.0625	-32.53	-31.00	-42.14	-41.10
Metric	0.25	-33.11	-31.12	-42.04	-37.77
	1.0	-33.17	**-32.05**	-40.02	-36.39
L_2	0.0625	-32.83	-30.47	-43.01	-42.96
Metric	0.25	-32.88	-31.12	-42.74	-42.83
	1.0	**-33.81**	-31.15	-42.95	-41.41
L_2	0.0625	-31.07	-29.33	-42.92	**-43.10**
Metric	0.25	-32.11	-29.52	**-43.01**	-42.93
Adaptive	1.0	-33.45	-30.57	-42.83	-42.29

Table VII.3: Results for GA-LS hybrids on the 2D conformation problem.

Table VII.2 shows the average performance of MC, MS and GA after 150000 function evaluations. Table VII.3 shows the average performance of the GA-LS hybrids using the local approximation methods.

The performance of the GA-LS hybrids is significantly better on the discrete angle representation. Note that the dynamics of the GA-LS hybrids differ on the two representations. On the angle representation, high local search frequencies are more efficient, though a statistical test did not reveal significant differences between the methods. On the discrete angle representation, low local search frequencies are more efficient. A statistical test did not reveal significant differences between the different GA-SW hybrids, but the GA-CG hybrids using the L_2 metric are significantly better than the GA-CG hybrids using fixed frequency and the inequality metric. Finally, the GA-LS hybrids are significantly better than MC, MS and GA.

These results are not directly comparable to those of Judson et al. [49] since they report the final results after 10^7 function evaluations. However, they report that the best solution found with their GA-CG hybrids is -44.3. After 150,000 function evaluations, the best solution found by these methods was -44.2. Further, when GA-LS hybrids were used with elitism the best solution found was -45.3, the global optimum! Both of these results require the discretized angle representation, which was not used by Judson et al. The best result for the GA-LS hybrids using the continuous angle representation was -35.9 after 150,000 function evaluations.

VII.C Drug Docking

One of the key elements of computer aided drug design is the docking of potential drug candidates to a target macromolecule. Manual methods of docking have been widely used [88]. They use sophisticated energy evaluations, but only allow the user to examine a limit number of docking conformations. The docking method described by Goodsell and Olsen [34] examines a large number of docking conformations automatically. This method uses simulated annealing (SA) to search the conformation space (see Section II.B.3) and performs rapid energy evaluations using molecular affinity potentials. Goodsell and Olson do not make a direct comparison with other automated docking methods like exhaustive search, but they note that methods like exhaustive search require simplified energy evaluations to avoid prohibitively expensive computational costs. They argue that the sophisticated search performed by SA enables them to use robust energy evaluations while maintaining reasonable computational costs.

The experiments in this section compare the performance of the GA and GA-LS hybrids to SA on a docking problem that models the docking of an inhibitor for HIV protease. Evaluations of the docking conformations were performed using the Autodock software developed by Olson et al. [70]. The conformation energy was evaluated using molecular affinity potentials, as described by Goodford [33]. The macromolecule is imbeded in a three-dimensional grid, and the energy of interaction is calculated for different atom types at every location of the grid. These energies are stored in tables that are used to rapidly compute the energy of a given conformation. Energies outside of grid have a default value of 1.0e5.

This docking problem has a total of 19 parameters. Three parameters specify the coordinates of the centroid of the molecule. The coordinates located on the grid are within the box defined by the points $[-9.401, -5.022, -15.038]$ and $[9.349, 13.728, 14.962]$. Four parameters specify the quarternion $(q_x, q_y, q_z), q_w$. The values q_x, q_y and q_z specify a unit vector (i.e. $q_x^2 + q_y^2 + q_z = 1$). This describes the direction in which the molecule will be rotated by q_w degrees. The remaining parameters specify the torsion angles of twelve rotatable bonds in the molecule, also in degrees.

The docking potential was minimized with SA using the Autodock software. Autodock starts SA using starting coordinates of the molecule at $(-2.0, 5.0, 7.7)$, the initial quarternion at $(1, 0, 0), 0$, and the initial torsion angles are uniformly generated. The initial temperature for SA is 5000.0, and the temperature is reduced by a factor of 0.85 every cycle, so after k cycles the temperature is $5000.0(0.85)^k$. SA was run for 50 cycles, each of which ran until 30,000 accepting states were found. To enable comparisons with the other methods, energy evaluations for conformations located outside the grid were counted as function evaluations.

To minimize the docking potential using the GA and GA-LS hybrids, the ranges of each of the parameters are normalized to $[0.0, 100.0]$. This normalization insures that mutations made to each dimension of the docking potential have the same chance of making changes of the same proportion. Preliminary experiments with GAs using unnormalized parameters had poor performance. Also, the quarternion parameters specifying the unit vector are not normalized to a unit length until the potential of each solution is evaluated. Thus, mutation treats all of the parameters uniformly. Finally, elitism is used in the GA and GA-LS hybrids.

Method	Freq	Average	Best
MC		184.5	19.7
GA		-28.6	-93.4
SA		-98.7	-107.9
GA-LS	0.0625	-94.7	-117.0
	1.0	**-108.4**	**-118.0**

Table VII.4: Results for docking problem.

Table VII.4 compares the performance of MC, GA, the GA-LS hybrids and SA after $1.5 \cdot 10^6$ function evaluations. The average is over 5 repeated trials. A statistical test of these results indicates that the GA-LS hybrids with frequency 1.0 are significantly better than MC. No other differences are significantly different. Figure VII.4 graphically compares the performance of these methods.

VII.D Summary and Discussion

The results of these applications demonstrate the relative efficiency of the GA-LS hybrids on a variety of different types of objective functions. In each of these applications, the GA-LS hybrids perform much better than MC, MS and the GA.

The results for the neural network problem illustrate the influence of the efficiency of the local search method. In particular, the comparison between GA-Batch-BP and GA-CG illustrates that GA-LS hybrids are not necessarily more efficient than the GA. Batch BP uses gradient information so inefficiently that the GA-Batch-BP hybrid has worse performance than the GA alone.

The experiments with the 2D molecular conformation problem confirm that the adaptive methods can significantly improve the performance of GA-LS hybrids. Further, these results provide evidence of the trade off between the reliability of the competitive selection and the refinement of the local search. Using the discrete representation constitutes an *a priori* bias since it focuses the GA's search on a subset of the entire conformation space. For this problem, this bias improves the performance

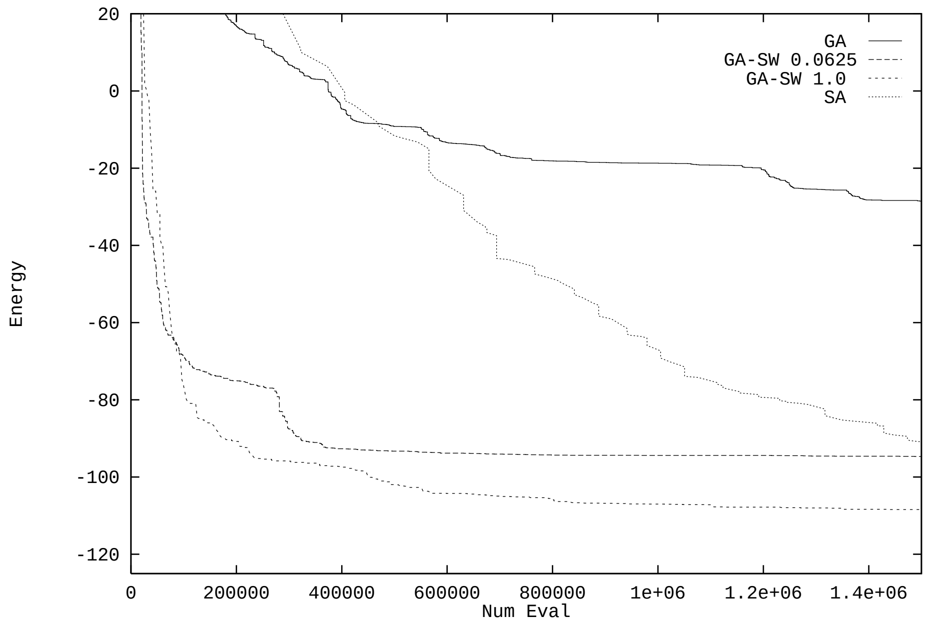


Figure VII.4: Optimization results for docking of inhibitor for HIV protease.

of the GA and, consequently, makes low frequencies of local search most efficient.

Finally, the results for the drug docking problem compare the efficiency of the GA-LS hybrids with SA. The GA-LS hybrids compare favorably, even after $1.5 \cdot 10^6$ function evaluations. Further, Figure VII.4 shows that the GA-LS hybrids perform much better at higher ϵ-accuracies. In fact, I expect that the performance of the GA-LS hybrids could be improved by performing local search with the neighborhood structure used in SA. The SA routine provided in Autodock performs a special transformation of the quarternion parameters when generating local neighbors. This transformation is probably more appropriate than the neighbors generated by the normal deviates used in Solis-Wets, so the comparison between the GA-LS hybrids and SA is conservative.

Chapter VIII

Conclusions

VIII.A Conclusions

The research in this dissertation has examined two general issues relating to GA-LS hybrids. First, I have addressed questions concerning the best use of local search with the GA and have described methods that provide significant improvement over standard GA-LS hybrids. The frequency of local search was the most influential parameter in our methods, and an analysis of GA-LS hybrids using a fixed frequency of local search provided considerable insight into the way the GA and local search algorithms interact. Our results indicate that the type of GA can influence the manner in which the local search algorithm should be used with the GA. For example, I have shown that my methods of reducing the local search frequency work particularly well with GSGAs, which have a many similar solutions in their populations.

I have also described parallel GSGAs that can efficiently utilize local search. These parallel GAs are a MIMD design of a GA initially implemented on SIMD architectures, where local search is difficult to apply selectively. The analysis of these parallel GSGAs indicates that they scale well to large populations, which is confirmed by an experimental analysis. These experiments recommend the asynchronous GSGAs since they perform well and have no synchronization penalties. MIMD GSGAs have been used with populations of up to 36,864 individuals, and I have argued that they can be used to tackle the same problems to which SIMD GSGAs have been applied. Further, I have noted that previous results indicate that these MIMD GSGAs will be competitive with other MIMD GAs.

GA-LS hybrids have also been used to solve problems in three different application domains: neural networks, molecular conformation and drug docking. These results confirm many of the observations made in the analysis of GA-LS hybrids. They confirm that using the adaptive methods can significantly improve the performance of GA-LS hybrids. A comparison is also made with simulated annealing, which demonstrates that GA-LS hybrids can perform better than simulated annealing, even after $1.5 \cdot 10^6$ function evaluations.

This dissertation has made two other technical contributions. First, I have analyzed the performance of probabilistic multistart. This analysis proves that selecting local search with a fixed probability is worse than the best of either MC or MS. This suggests that selectively applying local search will only be useful if either the global or local search method is used adaptively.

I have also described a generalization of the biological notion of the F statistic. The F statistic has been generalized to provide a measure of similarity for arbitrary distance metrics and has been related to the method of fitness sharing. The generalized F statistic is used by the distribution-based methods to selectively apply local search.

VIII.B Implications for Biological Models

Section II.C.3 noted that much of the inspiration for GA-LS hybrids comes from natural systems. Evolutionary algorithms like the GA take many cues from mechanisms observed in natural evolution, and local search is often equated with models of learning. Conversely, this research on artificial methods of adaptive search may have implications for models of evolution and learning in natural systems.

The superior performance of the GA-LS hybrids when compared with the

GA appears to provide confirmation of the "Baldwin Effect" [6, 40]. The experiments in Chapters V and VII indicate that learning can improve the efficiency of the GA. Consequently, GA-LS hybrids run for fewer generations than the GA. From a biological viewpoint, this can be viewed as an increased rate of evolution, which is a prime indication of a Baldwin effect. Note that the efficiency of GA-LS hybrids is improved even with simple local searches like Solis-Wets. This is important in natural systems where it is unlikely that additional information (like derivative calculations) is available to guide learning, but the local sampling of the type performed by Solis-Wets is not entirely unreasonable.

The experiments in Chapter VII demonstrate that there are many occasions in which GA-LS hybrids are most efficient when local search is applied infrequently. For example, low local search frequencies are more efficient when using large populations or when using elitist GAs which have strong selective pressure. These results suggest that the fitness of the entire population may be improved even when only a fraction of the population is applying learning methods.

The distribution-based methods of adapting the local search frequency are reminiscent of the effects of *inbreeding depression* [38], and may be useful for studying the effects of inbreeding on learning in natural systems. Inbreeding depression refers to the detrimental effects of inbreeding, which is indicated by a high F statistic for an individual. Now consider a GA-LS hybrid as a model of natural evolution and learning. By itself each individual in a population would naturally want to have a high probability of performing local search since that would maximize its fitness. However, the distribution-based methods lower an individual's local search frequency if its F statistic is high. This is analogous to the effect of inbreeding on phenotypic characteristics, where the local search frequency is viewed as a phenotypic characteristic of the individual.

Two differences between GA-LS hybrids and models of natural systems must be considered when examining the biological implications of the results in this dissertation. First, the fitness of individuals in natural systems is often *dependent* on the behavior of other individuals in the environment. In GA-LS hybrids, the fitness of individuals is *independent* of the fitness of other individuals in the population. As a result, the optimal solutions identified by a GA-LS hybrid may not correspond well with individuals in natural systems. Alternatively, populations of solutions that are optimal with respect to an independent fitness measure may be susceptible to invasion from nonoptimal solutions when evolved using a dependent fitness measure. In this case, the optimal solutions are not biologically plausible. This phenomenon was observed in Ackley and Littman's artificial life model [3]. Nowak and May observe similar phenomena in the context of game theory [69].

The second difference concerns the manner in which the rate of evolution is measured in natural systems and in GA-LS hybrids. There are three ways the cost of GA-LS hybrids can be evaluated. First, the cost of each generation is the sum of every operation applied to every individual (assuming a serial algorithm). When local search is applied infrequently, the cost of each generation decreases correspondingly. This is the most common method used in computational contexts. Second, the cost of the local search can be ignored completely, and the number of generations of the GA is used to measure the cost of the search. This measure has been used by Hinton and Nowlan [40] and Nolfi, Elman and Parisi [68]. Finally, the cost of each generation can be equated with the length of the longest local search performed in the population. Thus if local search is used by any individual in the population, there is no additional penalty incurred by allowing the rest of the population to perform local searches. This is probably the most biologically plausible cost measure, but it assumes a synchronous mating schedule which may not be a biologically plausible assumption.

VIII.C Future Directions

The methods that I have examined represent initial studies of some important areas of investigation. I describe some immediate extensions of this research and discuss some broader issues that need to be explored.

VIII.C.1 GA-LS Hybrids

I have considered a basic set of GA-LS hybrids that can be extended in a number of directions. First, additional methods of distribution- and fitness-based selection of local search may be of interest. The emphasis of this research is on distribution-based selection methods, but there are a variety of fitness-based methods that I have not considered here. As I noted earlier, most of the methods used to perform competitive selection in GAs can be used to perform fitness-based selection of local search. Furthermore, it is possible to combine distribution- and fitness-based methods, which I expect to combine the advantages of both.

The experimental results indicate that the type of GA used with local search can strongly influence the performance of the GA-LS hybrids. To simplify the analysis of the interactions between the GA and local search algorithms, relatively simple GAs were used in this dissertation. It is clear that further work needs to be done with GA-LS hybrids that use more sophisticated GAs. Methods like rank and truncation selection employ a stronger selective pressure, which focuses the GA's search more rapidly. Because of this stronger selection pressure, we expect that hybrids with these methods will require infrequent local search.

VIII.C.2 MIMD GSGAs

The most important extension of the results with MIMD GSGAs involves a comparison of this method with other MIMD GSGAs. Previous research with sequential models of parallel GAs indicates that MIMD GSGAs should be competitive with other MIMD GAs, but a direct comparison is needed to examine how the communication and synchronization costs affect the overall performance of MIMD GSGAs.

There are a number of other technical issues that are also worth investigating. First, the efficiency of the GSGAs may be improved by "pipelining" requests for data between the processors. Since only a fraction of the grid on each processor is communicated with its neighbors, it is possible to evaluate the individuals in the border regions first, perform communication with its neighbors, and then evaluate the rest of the individuals. This interleaving of communication and execution should offset some problems of load imbalance, and should improve even the asynchronous GSGA.

Another issue that needs to be addressed is the manner in which redundancy is handled in the GSGAs. The experiments with sequential GSGAs indicate that GSGAs have a lot of redundancy in the population. In preliminary experiments, I found that the redundancy was increased when the L_2 metric was used to reduce the local search frequency. The problem with this increased redundancy is that the cost of the selection mechanisms became the principle cost of each iteration of the algorithm. There are two ways that this problem could be handled. First, stopping conditions could be introduced that terminate the simulation once the redundancy in the population reaches a specified threshold. Second, the selection mechanism could be modified to avoid performing selection on local neighborhoods that contain a single solution. More research with these methods is needed to understand which of these is the better alternative.

Finally, I believe that there may interesting dynamics in the interaction between the shape of the GSGA's neighborhood structure and the shape of the partitions used to decompose the two-dimensional grid. For example, suppose strip partitions are used and let the GSGA's neighborhood be rectanglar. The height of the rectanglar neighborhood affects the size of the border regions communicated between processors. This has implications for the complexity of the GSGA, but I also expect this to affect the rate at which solutions are communicated between processors. Larger border regions should make it easier to communicate solutions between processors. Similarly, the width of the rectangular neighborhood affects the iteractions of individuals on the same processor. Very wide neighborhoods should facilitate the transmision of more optimal solutions to other parts of the processor's grid. Thus it appears that these two dimensions of the neighborhood structure can influence the inter-processor and intra-processor communication of solutions. The best balance between these two factors is not obvious, especially when non-square grids are used.

VIII.C.3 General Issues

The following are several general issues that are related to the current work and are worthy of investigation.

Constrained Optimization The focus of this dissertation has been on unconstrained global optimization. Several researchers have recently proposed methods that apply GAs to constrained optimization problems. A natural extension of these results would be to use GA-LS hybrids that combine GAs and local search methods that utilize constraint information. This approach seems promising for problems with highly nonlinear constraints that impose numerous local optima on the objective function.

Discrete Optimization These results offer an understanding for GA-LS hybrids that optimize functions defined on $\mathbf{R}^n$. I believe that the analysis of these GA-LS hybrids will generalize well to problems on a discrete domain. All of the mechanisms for selectively applying local search are independent of the search space, so they should be applicable to discrete problems. I still expect that it will be harder to generalize results from one discrete problem to another, but the discrete search space may make it easier to formally analyze the trade-off between competitive selection and local search.

Extended Applications The results with the molecular structure problems can be extended in a number of different directions. Scaling these problems to higher dimensions should be interesting. Since solutions to these problems are expensive to evaluate, sophisticated optimization methods are particularly important in higher dimensions. The reduced local search rate provided by the distribution-based methods should be particularly useful here. Also, conformation of more realistic molecules can be performed to evaluate the potential of this approach.

Appendix A

Generalizing the F Statistic

Consider a diploid individual with two chromosomes X and Y. These chromosomes can be decomposed into a sequence of alleles, $\{X_1, \ldots, X_n\}$ and $\{Y_1, \ldots, Y_n\}$. The biological notion of F statistic provides a metric for analyzing the similarity of X and Y with respect to a randomly mating population. When averaged over the entire population, the F statistic provides a measure of the *inbreeding*, or homogeneity of the population.

The biological definition of the F statistic simply measures the average number of points at which X and Y differ. We generalize the definition of the F statistic to allow for other distance measures between two chromosomes. As a consequence, the generalized F statistic can be computed for chromosomes on any space for which a distance metric is available.

A.A Formalism

Hartl [38] defines the inbreeding coefficient of an individual (relative to the total population) to be

$$F_{IT} = \frac{H_T - H_I}{H_T}$$

where

- H_T - the expected heterozygosity of an individual in an equivalent random mating total population

- H_I - the heterozygosity of an individual

H_I can be interpreted as either the average heterozygosity of all of the genes in an individual or as the probability of the heterozygosity of any one gene. Let

$$\delta(a_i, a_j) = \begin{cases} 0, & a_i = a_j \\ 1, & \text{otherwise} \end{cases}$$

where a_i and a_j are two values of alleles. Then we can define the difference between two sets of alleles as

$$D_1(X, Y) = \frac{1}{n} \sum_{i=1}^{n} \delta(X_i, Y_i) \equiv H_I$$

Note that this equation is independent of the cardinality of the set of alleles which are contained in the individual. The only thing that matters is that the alleles are different.

The value of H_T can be formulated using D_1. Let P be a distribution over the space of possible sets of genes, G. Then

$$H_T = \int_G \int_G D_1(X, Y) dP(X) dP(Y)$$

The dependency on the distribution P makes the F statistic relative to the level of inbreeding that was present in the initial population.

A.B Generalization

The distance measure D_1 provides a notion of distance that depends on whether the individual alleles in X and Y are different. If we consider alleles that assume values in an arbitrary space, we may have a notion of distance between the alleles. In these contexts, the measure D_1 will be unnecessarily crude. Further, D_1 will be inappropriate when $|G|$ is not finite.

Note that D_1 is a metric on G. Given a metric D_2 on a different space of genotypes, G', we can use the formulas for H_T and H_I to compute a generalized F-statistic. The difficult part of using the definitions in a general setting is the computation of H_T. For example, let $G' = [A, B]^n$ and let D_2 be the L_2 norm

$$D_2(X, Y) = \|X - Y\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

then the computation of H_T is very difficult, even when P is the uniform distribution.

We now examine two metrics that can be analyzed when P is the uniform distribution. These metrics will be useful when estimating the F-statistics of individuals in GA's with floating point encodings. The initial population of the GA is typically created by sampling from a uniform distribution over G, so these F-statistics will be appropriate.

Let $G' = [A, B]^n$, and let D_2 be the L_1 norm

$$D_2(X, Y) = \|X - Y\|_1 = \sum_{i=1}^{n} |x_i - y_i|,$$

then

$$H_T = \frac{n}{3}(B - A)$$

if P is uniform. If D_2 is the squared L_2 norm

$$D_2(X, Y) = \|X - Y\|_2^2 = \sum_{i=1}^{n} (x_i - y_i)^2,$$

then

$$H_T = \frac{n}{3}(B - A)^2$$

if P is uniform.

Similarly, let $G' = \{A, \ldots, B\}^n$. If D_2 is the L_1 norm then

$$\frac{n(B - A)(B - A + 2)}{3(B - A + 1)}$$

If D_2 is the squared L_2 norm then

$$\frac{n(B - A)(B - A + 2)}{6}$$

The generalized F statistic assumes values less than or equal to one. The F statistic is zero if the distance between the chromosomes is equal to H_T

Appendix B

Analytic Gradients for the 2D Conformation Problem

This appendix describes the equations used to analytically calculate the derivative for the simple conformation problem consider by Judson et al. [49].

$$f(\alpha) = \sum_{i<j} \left[\epsilon \left(\frac{r^*}{r_{ij}} \right)^{12} - 2\epsilon \left(\frac{r^*}{r_{ij}} \right)^{6} \right]$$

where r_{ij} is the interatom distance

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

The angles α are used to calculate the coordinates x and y. Note that the angles begin with α_1, but we add $\alpha_0 = 0$. Let

$$\Theta_k = \sum_{i=1}^{k} \alpha_i$$

then $(x_0, y_0) = (0, 0)$, and

$$x_i = \sum_{k=0}^{i-1} \cos(\Theta_k)$$

$$y_i = \sum_{k=0}^{i-1} \sin(\Theta_k)$$

We use the chain rule to calculate $\frac{\partial f}{\partial \alpha_m}$.

$$\frac{\partial f}{\partial r_{ij}} = -12\epsilon \sum_{i<j} \left[\left(\frac{(r^*)^{12}}{r_{ij}^{13}} \right) - \left(\frac{(r^*)^{6}}{r_{ij}^{7}} \right) \right] \frac{\partial r_{ij}}{\partial \alpha_m}$$

$$\frac{\partial r_{ij}}{\partial \alpha_m} = \frac{1}{r_{ij}}\left[(x_i - x_j)\frac{\partial(x_i - x_j)}{\partial \alpha_m} + (y_i - y_j)\frac{\partial(y_i - y_j)}{\partial \alpha_m}\right]$$

Assuming that $j > i$, we have

$$\frac{\partial(x_i - x_j)}{\partial \alpha_m} = -\sum_{k=i}^{j-1}\frac{\partial}{\partial \alpha_m}\cos(\Theta_m) = \sum_{k=\max(i,m)}^{j-1}\sin(\Theta_k)$$

$$\frac{\partial(y_i - y_j)}{\partial \alpha_m} = -\sum_{k=i}^{j-1}\frac{\partial}{\partial \alpha_m}\sin(\Theta_k) = -\sum_{k=\max(i,m)}^{j-1}\cos(\Theta_k)$$

Now we can redefine $\frac{\partial r_{ij}}{\partial \alpha_m}$,

$$\frac{\partial r_{ij}}{\partial \alpha_m} = \frac{1}{r_{ij}}\left[(x_i - x_j)(\sum_{k=\max(i,m)}^{j-1}\sin(\Theta_k)) + (y_i - y_j)(-\sum_{k=\max(i,m)}^{j-1}\cos(\Theta_k))\right]$$

and observe that this is zero when $m \leq i$. Assuming that $m > i$, note that

$$-\sum_{k=m}^{j-1}\cos(\Theta_k) = (x_m - x_j)$$

$$\sum_{k=m}^{j-1}\sin(\Theta_k) = -(y_m - y_j)$$

Thus

$$\frac{\partial r_{ij}}{\partial \alpha_m} = \frac{1}{r_{ij}}\left[(x_i - x_j)(y_j - y_m) + (y_i - y_j)(x_m - x_j)\right]$$

Combining terms, we get

$$\frac{\partial f}{\partial \alpha_m} = -12\epsilon\sum_{i=0}^{m-1}\sum_{j=m+1}^{n}\left[\left(\frac{(r^*)^{12}}{r_{ij}^{14}}\right) - \left(\frac{(r^*)^{6}}{r_{ij}^{8}}\right)\right]\left[(x_i - x_j)(y_j - y_m) + (y_i - y_j)(x_m - x_j)\right]$$

Bibliography

[1] David H. Ackley. *A Connectionist Machine for Genetic Hillclimbing.* Kluwer Academic Publishers, 1987.

[2] David H. Ackley. A case for Lamarackian evolution. In *To appear in Proceedings of the Third Conf. on Artificial Life*, 1993.

[3] David H. Ackley and Michael L. Littman. A video presentation of "learning from natural selection in an artificial environment". In Chris G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Video Proceedings of the Second Conference on Artificial Life*, pages 487–509. Addison-Wesley, 1990.

[4] Thomas Bäck and Frank Hoffmeister. Extended selection mechanisms in genetic algorithms. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 92–99. Morgan-Kaufmann, 1991.

[5] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 2–9. Morgan-Kaufmann, 1991.

[6] Richard K. Belew. Evolution, learning, and culture: Computational metaphors for adaptive algorithms. *Complex Systems*, 4(1):11–49, 1990.

[7] Richard K. Belew, John McInerny, and Nicol N. Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. In Chris G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Proceedings of the Second Conference on Artificial Life*, pages 511–548. Addison-Wesley, 1991.

[8] C.G.E. Boender and A.H.G. Rinnooy Kan. Bayesian stopping rules for multistart global optimization methods. *Mathematical Programming*, 37:59–80, 1987.

[9] Heinrich Braun. On solving the travelling salesman problems by genetic algorithms. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 129–133. Springer-Verlag, 1990.

[10] Matthew Clark, Richard D. Cramer, III, and Nicole Van Opdenbosch. Validation of the general purpose Tripos 5.2 force field. *Journal of Computational Chemistry*, 10(8):982–1012, 1989.

[11] N.E. Collins, R.W. Eglese, and B.L. Golden. Simulated annealing - an annotated bibliography. *American Journal of Mathematics and Management Sciences*, 8(3 & 4):209–307, 1988.

[12] Robert J. Collins and David R. Jefferson. Selection in massively parallel genetic algorithms. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 249–256, 1991.

[13] J.S. Cramer. *The Logit Model: An introduction for economists*. Edward Arnold, 1991.

[14] Yuval Davidor, Takeshi Yamada, and Ryohei Nakano. The ECOlogical framework II: Improving GA performance at virtually zero cost. In Stephanie Forrest, editor, *Proceedings of the Fifth Intl. Conf. on Genetic Algorithms*, pages 171–176. Morgan-Kaufmann, 1993.

[15] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[16] Kenneth A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, 1975.

[17] C.C.Y. Dorea. Limiting distribution for random optimization methods. *SIAM Journal of Control and Optimization*, 24(1):76–82, 1986.

[18] C.C.Y. Dorea. Stopping rules for a random optimization method. *SIAM Journal of Control and Optimization*, 28(4):841–850, 1990.

[19] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[20] I. Fiodorova. Search for the global optimum of multiextremal problems. In *Optimal Decision Theory 4*, pages 93–100, Lithuanian SSR Acad. of Sci., 1978. Inst. of Math. and Cybern.

[21] Christodoulos A. Floudas and Panos M. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*, volume 455 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[22] David R. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1994.

[23] Stephanie Forrest and Melanie Mitchell. The performance of genetic algorithms on Walsh polynomials: Some anomalous results and their explanations. In *Proceedings of the 4th conference on Genetic Algorithms*, June 1991.

[24] Michael R. Garey and David S. Johnson. *Computers and Intractability - A guide to the theory of NP-completeness.* W.H. Freeman and Co., 1979.

[25] John Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal of Computation*, 6(4):675–695, 1977.

[26] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical optimization.* Academic Press, 1981.

[27] David Goldberg. The theory of virtual alphabets. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 13–22. Springer-Verlag, 1990.

[28] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 301–315. Morgan-Kauffmann, 1991.

[29] David E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[30] D.E. Goldberg. Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, 3:129–152, 1989.

[31] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Co., Inc., 1989.

[32] S. Gomez and A. V. Levy. *The tunneling method for solving the constrained global optimization problem with several non-connected feasible regions*, pages 34–47. Lecture Notes in Mathematics. Springer-Verlag, 1982.

[33] P. J. Goodford. A computational procedure for determining energetically favorable binding sites on biologically important molecules. *J. Med. Chem.*, 28:849–857, 1985.

[34] David S. Goodsell and Arthur J. Olson. Automated docking of substrates to protiens by simulated annealing. *Protiens: Structure, Function and Genetics*, 8:195–202, 1990.

[35] V. Scott Gordon and Darrell Whitley. Serial and parallel genetic algorithms as function optimizers. In Stephanie Forrest, editor, *Proceedings of the Fifth Intl. Conf. on Genetic Algorithms*, pages 177–183. Morgan-Kaufmann, 1993.

[36] William E. Hart and Richard K. Belew. Optimizing an arbitrary function is hard for the genetic algorithm. In *Proceedings of the 4th conference on Genetic Algorithms*, pages 190–195, June 1991.

[37] William E. Hart and Richard K. Belew. Optimization with genetic algorithm hybrids that use local search. In *Plastic Individuals in Evolving Populations*, 1994. (to appear).

[38] Daniel L. Hartl. *A primer on population genetics.* Sinauer Associates, 1981.

[39] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Lecture Notes Volume 1, Sante Fe Institute, Studies in the Sciences of Complexity. Addison-Wesley, 1991.

[40] Geoffrey E. Hinton and Steven J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[41] Frank Hoffmeister and Thomas Bäck. Genetic algorithms and evolutionary strategies: Similarities and differences. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 455–469. Springer-Verlag, 1990.

[42] John H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1976.

[43] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Pub. Co., 1979.

[44] Lester Ingber. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12(8):967–973, 1989.

[45] Lester Ingber and Bruce Rosen. Genetic algorithms and very fast simulated reannealing - a comparison. *Mathematical and Computer Modelling*, 16(11):87–100, 1992.

[46] Cezary Z. Janikow and Zbigniew Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 31–36. Morgan-Kaufmann, 1991.

[47] David S. Johnson. Local optimization and the traveling salesman problem. In M.S. Paterson, editor, *Automata, Languages and Programming - 17th International Colloquium*, pages 446–461, New York, July 1990. Springer-Verlag. Volume #443.

[48] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of computer and system sciences*, 37(1):79–100, Aug 1988.

[49] R.S. Judson, M.E. Colvin, J.C. Meza, A. Huffer, and D. Gutierrez. Do intelligent configuration search techniques outperform random search for large molecules? *International Journal of Quantum Chemistry*, pages 277–290, 1992.

[50] Ron Keesing and David G. Stork. Evolution and learning in neural networks: The number and distribution of learning trials affect the rate of evolution. In Richard P. Lippmann, John E. Moody, and David S. Touretzky, editors, *NIPS 3*, pages 804–810. Morgan Kaufmann, 1991.

[51] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[52] Scott R. Kohn and Scott B. Baden. A robust parallel programming model for dynamic non-uniform scientific computations. In *Proceedings of the 1994 Scalable High Performance Computing Conference*, May 1994.

[53] Clyde P. Kruskal and Alan Weiss. Allocating independent subtasks on parallel processors. In *ICCP 1984*, pages 236–240, 1984. Extended Abstract.

[54] Scott M. Le Grand and Kenneth M. Merz, Jr. The application of the genetic algorithm to the minimization of potential energy functions. *Journal of Global Optimization*, 3:49–66, 1993.

[55] R.S. Maier, J.B. Rosen, and G.L. Xue. A discrete-continuous algorithm for molecular energy minimization. Unpublished manuscript, Mar 1992.

[56] John M.N. McInerny. *Biologically Influenced Algorithms and Parallelism in Non-Linear Optimization.* PhD thesis, University of California, San Diego, 1992.

[57] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs.* Springer-Verlag, 1992.

[58] Melanie Mitchell, Stephani Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Toward a practice of autonomous systems. Proceedings of the First European Conference on Artificial Life*, pages 245–54. MIT Press, 1992.

[59] David J. Montana and Lawrence Davis. Training feedforward neural networks using genetic algorithms. In *IJCAI 1989*, pages 762–767, 1989.

[60] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–85, 1988.

[61] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 271–278. Morgan-Kaufmann, 1991.

[62] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17:619–632, 1991.

[63] Heinz Mühlenbein. Evolution in time and space - the parallel genetic algorithm. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337. Morgan-Kauffmann, 1991.

[64] Peter Neuhaus. Solving the mapping-problem - experiences with a genetic algorithm. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 170–175. Springer-Verlag, 1990.

[65] Harald Niederreiter. A quasi-Monte Carlo method for the approximate computation of the extreme values of a function. In Paul Erdös, editor, *Studies in Pure Mathematics*, pages 523–529. Birkhäuser Verlag, 1983.

[66] Harald Niederreiter. Quasi-Monte Carlo methods for global optimization. In W. Grossmann, G. Pflug, I. Vincze, and W. Wertz, editors, *Proceedings of the 4th Pannonian Symposium on Mathematical Statistics*, pages 251–267. Bad Tatzmannsdorf, Austria, 1983.

[67] Harald Niederreiter and Paul Peart. Localization of search in quasi-Monte Carlo methods for global optimization. *SIAM Journal of Scientific and Statistical Computing*, 7(2):660–664, April 1986.

[68] Stefano Nolfi, Jeffrey L. Elman, and Domenico Parisi. Learning and evolution in neural networks. Technical Report CRL 9019, Center for Research in Language, University of California, San Diego, July 1990.

[69] M. A. Nowak and R. M. May. The spatial dilemas of evolution. *Intl. Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, 3(1):35–78, 1993.

[70] Arthur J. Olson, David S. Goodsell, Garrett M. Morris, and Ruth Huey. *Autodock User Guide.* Scripps Research Institute, Department of Molecular Biology, 1994.

[71] Christos H. Papadimitriou, Alejandro A. Schäffer, and Mihalis Yannakakis. On the complexity of local search. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 438–45, 1990.

[72] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization - Algorithms and Complexity.* Prentice Hall, Inc., 1982.

[73] A.T. Phillips and J.B. Rosen. A computation comparison of two methods for constrained global optimization. Unpublished manuscript, 1992.

[74] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipies in C - The Art of Scientific Computing.* Cambridge University Press, 1990.

[75] L. A. Rastrigin. *Systems of extremal control.* Nauka, 1974.

[76] A.H.G. Rinnooy Kan, C.G.E. Boender, and G.Th. Timmer. A stochastic approach to global optimization. In K. Schittkowski, editor, *Computational Mathematical Programming*. NATO ASI Series, Bol. F15, 1985.

[77] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods - part I: Clustering methods. *Mathematical Programming*, 39:27–56, 1987.

[78] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods - part II: Multi level methods. *Mathematical Programming*, 39:57–78, 1987.

[79] Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[80] David E. Rumelhart, Geoffrey E. Hinton, and James L. McClelland. A general framework for parallel distributed processing. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing*, volume 1, pages 45–76. MIT Press, 1986.

[81] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, 1986.

[82] J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, and Rajarshi Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. David Schaffer, editor, *Proceedings of the Third Intl. Conf. on Genetic Algorithms*, pages 51–60. Morgan-Kaufmann, 1989.

[83] Nicol N. Schraudolph and Richard K. Belew. Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9:9–21, 1992.

[84] F.J. Solis and R.J-B. Wets. Minimization by random search techniques. *Mathematical Operations Research*, 6:19–30, 1981.

[85] William M. Spears and Kenneth A. De Jong. An analysis of multi-point crossover. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 301–315. Morgan-Kauffmann, 1991.

[86] William M. Spears and Kenneth A. De Jong. On the virtues of parametrized uniform crossover. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 230–236. Morgan-Kaufmann, 1991.

[87] Piet Spiessens and Bernard Manderick. A massively parallel genetic algorithm: Implementation and first analysis. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 279–285, 1991.

[88] N. J. Stedmann, G. M. Morris, and P. J. Atkinson. Bibliography of theoretical calculations in molecular pharmacology. *J. Mol. Graphics*, 5:211–222, 1987.

[89] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, 1989.

[90] Larry E. Toothaker. *Multiple Comparisons for Researchers*. Sages Publications, 1991.

[91] Aimo Törn and Antanas Žilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[92] Craig A. Tovey. Hill climbing with multiple local optima. *SIAM Journal of Algorithms and Discrete Mathematics*, 6(3):384–393, 1985.

[93] Craig A. Tovey. Low order polynomial bounds on the expected performance of local imporvement algorithms. *Mathematical Programming*, 35:193–224, 1986.

[94] J. F. Traub, G. W. Wasilkowski, and H. Woźniakowski. *Information-Based Complexity*. Academic Press, Inc., 1988.

[95] Nico L.J. Ulder, Emile H.L. Aarts, Hans-Jürgen Bandelt, Peter J.M. van Laarhoven, and Erwin Pesch. Genetic local search algorithms for the traveling salesman problem. In Hans-Paul Schwefel and Reinhard Männer, editors, *Parallel Problem Solving from Nature*, pages 109–116. Springer-Verlag, 1990.

[96] Gregor von Laszewski. Intelligent structural operators for the k-way graph partitioning problem. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 45–52. Morgan-Kaufmann, 1991.

[97] Michael D. Vose and Gunar E. Liepens. Schema disruption. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 237–242. Morgan-Kaufmann, 1991.

[98] E.D. Weinberger. Fourier and Taylor serics on fitness landscapes. *Biological Cybernetics*, 65:321–330, 1991.

[99] Halbert White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1(4):425–464, 1989.

[100] D. Whitley, K. Mathias, and P. Fitzhorn. Delta coding: An iterative search strategy for genetic algorithms. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 77–84. Morgan-Kaufmann, 1991.

[101] Darrell Whitley. Cellular genetic algorithms. In Stephanie Forrest, editor, *Proceedings of the Fifth Intl. Conf. on Genetic Algorithms*, page 658. Morgan-Kaufmann, 1993.

[102] Darrell Whitley, Stephen Dominic, and Rajarshi Das. Genetic reinforcement learning with multilayer neural networks. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms*, pages 562–569. Morgan-Kaufmann, 1991.

[103] Alden H. Wright. Genetic algorithms for real parameter optimization. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 205–218. Morgan-Kauffmann, 1991.