Search Strategies for Global Optimization

Megan Hazen

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2008

Program Authorized to Offer Degree: Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Megan Hazen

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

Maya Gupta

Reading Committee:

_____

Howard Chizeck

_____

Maya Gupta

_____

Zelda Zabinsky

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Search Strategies for Global Optimization

Megan Hazen

Chair of the Supervisory Committee:
Assistant Professor Maya Gupta
Department of Electrical Engineering

Global optimizers are powerful tools for searching complicated function spaces such as those found in modern high-fidelity engineering models. These models provide increasingly detailed insights into system behaviors, but are often expensive to evaluate and difficult to search. While techniques exist for solving global optimization problems there is still room for developing faster, more reliable, and easier to implement algorithms. This thesis investigates the design of a global optimization algorithm, with a focus on the search strategies of gradient descent, memory, and multiresolution search. Major contributions presented in this thesis include the proposal of a regional gradient, the examination of the use of regional gradients in global search, and the proposal and development of a novel optimization algorithm. Additionally, a case study using the multiresolution estimated gradient architecture (MEGA) algorithm to solve a classification problem is presented.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author would like to thank Maya Gupta for her superb advisory work, and the rest of the supervisory committee for this thesis: James Burke, Howard Chizeck, Dan Grossman, and Zelda Zabinsky for their contributions. Additionally the author would like to the thank the staff of the Applied Physics Laboratory at the University of Washington for their support, particularly Warren Fox, Bob Miyamoto, Bob Odom, Dian Gay. Special appreciation is due to Luca Cazzanti, Andy Ganse, Julia Hsieh, and Kevin Jamieson for their feedback, patient explanations, and occasionally code. Finally, the author expresses deep gratitude to her family, especially Joshua Buergel, for sticking around every step of the way.

This thesis is dedicated to my grandparents: Ed and Alice Berrier, Cliff and Mary Hartman, Allen and Edith Hazen, and Richard and Francie Shuey, who paved the way for me in countless ways. I hope to make them proud.

Chapter 1

GLOBAL OPTIMIZATION

This thesis investigates the design of a global optimization algorithm, with a focus on the search strategies of gradient descent, memory, and multiresolution search. Global optimizers are powerful tools for searching complicated function spaces such as those found in modern high-fidelity engineering models. These models provide increasingly detailed insights into system behaviors, but are often expensive to evaluate and difficult to search. While techniques exist for solving global optimization problems there is still room for developing faster, more reliable, and easier to implement algorithms. Major contributions presented in this thesis include the proposal of a regional gradient, the examination of the use of regional gradients in global search, the proposal and development of a novel optimization algorithm. Additionally, a case study using the multiresolution estimated gradient architecture (MEGA) algorithm to solve a classification problem is presented.

I begin by describing the global optimization problem and reviewing some existing techniques and ideas. The body of this research is the development and analysis of a new optimization algorithm. The MEGA algorithm, introduced in Chapter 2, is motivated by successful search strategies in existing optimization algorithms including estimated gradients, memory, and multiresolution search. MEGA shows promise to perform well on structured global problems and to scale well with increasing search space dimensionality. A new concept, the regional gradient, is proposed in Chapter 3 as a tool for analyzing and improving global optimization algorithms. The regional gradient is used to look at the impact of different regional gradient estimates on global search performance. Chapter 4 of this thesis looks at the performance of MEGA in application. Conclusions and some open questions are discussed in Chapter 5.

1.1 Background

Consider a search for a global optima of an objective function $f(x)$ defined on some compact set $\mathcal{S} \subset \mathcal{R}^D$. The optimization problem is to find a global minimum x^* of an objective function $f(x)$ such that

$$x^* = \underset{x \in \mathcal{S}}{\operatorname{argmin}} f(x). \tag{1.1}$$

Functions may also have local minima, where the local minimum x_p^* of some region $\mathcal{S}_p \subset \mathcal{S}$ solves

$$x_p^* = \underset{x \in \mathcal{S}_p}{\operatorname{argmin}} f(x).$$

Complex engineering models are the types of objective functions $f(x)$ with which this research is primarily concerned. These problems often have functions with some structure, such that using more information about the function will result in a more efficient search. However, in many problems the objective function $f(x)$ is modeled as a black box: $f(x)$ can only be obtained for a specific x by running a program, taking measurements, or modeling a system. Functional information, in particular the function gradient, is unobtainable, yet estimates of gradients based on previously-evaluated operating points may be useful. These models are also extremely complex and may have many local optima, non-linear behavior, and even discontinuities.

One of the challenges is to find an algorithm that is suited to the specific type of objective function being optimized. No algorithm is ideally suited for all types of problems. Those algorithms that are suited to the most unstructured problems tend to be less efficient for the highly structured problems. The first step, therefore, is to understand the type of problem at hand. A summary of problem aspects to consider may be found in [58]. These aspects include the size of the region of attraction around the global minimum, the number of global minimizers, whether the global minimum is embedded, and finally, the affordable number of function evaluations.

A global minimum's area of attraction is the region from where a local search will find that global minimum. If there is more than one global minimizer, the region of attraction is the union of the regions around all the global minimizers. In global optimization a global

search is needed to find promising regions of attraction, and then local searches may be used to hone in on the precise minimum.

The notion of an embedded minimum is that the global minimum is surrounded by local minima, so that finding these local minima may put the search in a region of attraction for the global minimum. An example of an embedded global minimum may be found in Figure (1.1). Objective functions that have some structure to them, such as an embedded minimum, allow for the use of optimizing algorithms that capitalize on that structure.

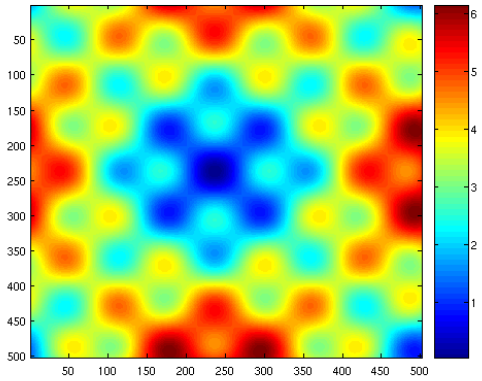

Figure 1.1: The sinusoidal test function. This is an example of an embedded global minimum.

Another aspect of optimization problems that poses many challenges is high dimensionality. Not only does this increase the volume of space that the optimizing algorithm needs to search, but a higher percentage of that space is near the edges of the search domain. Many algorithms that perform successfully in low dimensions begin to falter at higher dimensions. Engineering models frequently require the optimization of several, if not hundreds, of parameters.

Given this great variety in objective function characteristics an added challenge is evaluating how well optimization functions perform. In local optimization there is theory to prove convergence of algorithms, and some effort has been made to do this for global optimization algorithms. There are probabilistic convergence guarantees for pure and adaptive random search methods [65], and some attempts have been made to compare other stochastic algo-

rithms theoretically [52]. In practice, the most useful information about global optimization algorithm performance is arguably obtained through computational assessments.

Computational analysis of algorithms can be time-consuming and difficult. Most algorithms are tested on one precise type of problem so it can be difficult to generalize to other types of objective functions. Additionally, since the performance measures used to evaluate the optimization algorithms are inconsistent, it is difficult to compare one algorithm to another.

In most standardized tests the measure of performance is either the best value found during a series of optimization runs, the number of function evaluations in a successful optimization run, or some combination of the two. There are also a number of functions designed to test optimization algorithms. Each function offers its own challenges, but no function can serve as the only test for an algorithm. There have been some attempts to collect test functions and standardize algorithmic assessment, as well as proposals for concise performance metrics [12].

The choice of performance measure can not be entirely standardized because some users, such as those interested in real-time control, may wish for a feasible answer in a short amount of time. Other users, such as those designing structural components, may require a precise global optimum and have fewer time limits. Ideally optimization algorithms are evaluated in a variety of testing situations and enough information is provided about performance aspects for the users to choose the appropriate algorithm [27].

Attributes of a good numerical optimization algorithm include generality, efficiency, reliability, and ease of use. Generality refers to the variety of objective functions for which the algorithm is suited. Efficiency measures the amount of time (or how few function evaluations) are required to converge to a suitable answer. Reliability refers to the percentage of optimization runs that are able to find a suitable solution. Finally, maximizing ease-of-use requires both minimizing the effort to implement the algorithm, and minimizing the amount of tuning needed to apply the algorithm. In this research global optimization is examined with an eye towards each of these four criteria.

1.2 Previous Work

Optimization is an important tool in today's world. The advent of computers means that a wide variety of problems may be partially addressed, or solved, with an optimization approach. Optimization is such an important tool that there is a vast body of work associated with it. In this research a new algorithm is developed using the three strategies of gradient descent, multiresolution searching, and memory. We will start with a brief background summary of optimization and then focus on the way previous work contributes to our knowledge of these three specified concepts.

Optimization research can be divided into separate categories with the most obvious division separating local and global optimization. Local optimization refers to searching convex or unimodal functions. There are a number of approaches to local optimization, documented in numerous sources such as [42, 44]. There is also vital research going on to solve more difficult linear systems and inverse problems. This work may also be useful for solving complicated multi-modal problems, if some assumptions about the function hold true. However, in this research we are focused on multi-modal functions which are modelled as black-boxes. Traditional local optimization is not appropriate. Therefore, in this section the focus is prior work on global optimization.

Within global optimization there are a number of approaches. There are examples of deterministic searches (such as grid searches [18]), but many of the most popular algorithms have stochastic elements. Pure random search (PRS) [64], in which each point is drawn randomly from the search space and evaluated, is on the other end of the scale. PRS has been proven to have probabilistic convergence, and is perhaps the only way to search a function with no structure. Since most functions have some sort of structure (for example, some smoothness constraint) many algorithms capitalize on some deterministic movement combined with some stochastic behavior.

A variant of PRS is controlled random search (CRS) [47], in which randomly generated operating points are combined with a deterministic method for determining the next point to be evaluated. CRS uses a simplex of points to generate a new operating point. Another example of an algorithm that has some deterministic movement (a gradient descent step)

and some stochastic movement (an annealing mechanism) has been offered in [63]. It is also worth recognizing that the extremely popular method of multi-start (in which a possibly deterministic algorithm is run many times with different randomly generated starting conditions) injects randomness every time it is applied. Multi-start algorithms are powerful because each starting point has a new chance to be within the area of attraction of the global minimum.

There is another bifurcation in global optimization algorithms between gradient-based and gradient-free searches. There are many algorithms that use gradients, or estimated gradients, in some form. These algorithms are discussed below. Gradient-free searches, also known as direct search algorithms, use only information at hand. Random search, genetic algorithms, and pattern searches are all types of direct search algorithms. An example of direct search includes simulated annealing (SA) [28]. Originally proposed in 1983, SA is one of the workhorses of the global optimization community. The advantage that SA has over greedy random searches is that it occasionally accepts operating points with worse functional values. This hill-climbing behavior allows SA to escape from local minima and search the global space. PRS and CRS, described above, are also examples of direct search algorithms.

Another class of gradient-free problems uses pattern searching to generate new points. Pattern searching refers to the method of generating each new operating point by searching in a pre-designated pattern around the current operating point or points [29, 22]. The pattern provides a set of trial points to be evaluated. In compass search, for example, a new point is evaluated one step away from the current point in each of the cardinal directions. A new point is accepted if it has a smaller function value than the current operating point. If no improving point is found with any of the poll points the search can be refined by modifying the stepsize. It has been proven that algorithms with this basic scheme converge to local minima with only a few restrictions [57]. Variations on the basic pattern search include accepting non-improving points, the manner in which the poll points are generated, and the mechanism for refining the search. Further work has been done to generalize pattern search for mixed variable problems, and to allow for a search step to refine the results of a poll step [4].

One great advantage of pattern searches is that they may be used for discrete problems, and also for objective functions in which the output is non-numeric. Since new points are accepted if they are improving on the old point, the only feedback needed is whether a new point is better or worse.

Similar to the pattern search is the Nelder-Mead simplex method. First proposed in [38] this method uses a set of points to generate a new polling point in what is expected to be a down-hill direction. A simplex in D dimensions is formed with $D + 1$ linearly independent points (such as a triangle on a plane). In this method the worst performer of the simplex points is 'reflected' through the simplex to generate a polling point. If the new point performs well the reflection may be extended to attempt a larger step, and if it performs poorly the reflection may be shrunk to take a smaller step. The Nelder-Mead method is interesting because it is a direct search method that also attempts to move in a downhill direction. However, it has many of the same problems that generalized pattern search does and can get trapped in local minima.

The idea of a simplex is also found in many other places, such as in CRS. In CRS the reflected point is chosen randomly, so there is no concept that the reflection will direct the search downhill. The Nelder-Mead method of reflection, however, is straight-forward enough that it can be applied in a myriad of situations.

There are many reasons to use the direct search algorithms described above. They are easy to implement, place few constraints on the objective function, and can yield good results. They have some drawbacks however, such as slow asymptotic convergence, poor scaling with dimensionality, and in some case, only local convergence proofs [29].

A final class of algorithms to consider is those that address high-dimensional objective functions. These algorithms are important because problems with high dimensional input always pose difficulty. In this research the algorithm design attempts to scale reasonably with increasing dimension by generating search direction based on all dimensions. However, there have been a number of attempts to address high-dimensionality with different approaches. One common one is to divide the space into sub-spaces and search in a smaller dimensionality at a time [59, 40].

The following discussion will focus on how prior work uses gradient descent, multireso-

lution search, and memory. The work found below may also fit into one of the broad classes described above, but the focus is on how it contributes to our understanding of the three issues.

1.2.1 Gradient Descent

One of the algorithm components proposed in this research is a gradient descent step. Gradient methods of optimization have been around since the mid nineteenth century. All gradient-based algorithms use the first partial derivatives of the objective function in their computation

$$g(x) = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} ... \frac{\partial f}{\partial x_n}],$$

and that they are governed by an equation of the form

$$x_{new} = x_{old} - \sigma g(x).$$

Gradient techniques differ in the stepsizes (σ) used, and in cases where the objective function is not analytically differentiable, the gradients may be approximated or estimated [44].

One well known traditional gradient descent method is the Newton-Raphson method. In this method both the first and second derivatives are used as in the following (one-dimensional) formula:

$$x_{new} = x_{old} - \frac{g(x_{old})}{g'(x_{old})}.$$

An advantage is that the stepsize is optimally based on how quickly the gradient is changing, a value represented by the Hessian matrix of second derivatives. A disadvantage is that calculating the second derivative may be impossible, and is certainly computationally intensive for high dimensions [42].

In many cases there is no analytical expression for $g(x)$ and the gradient must be estimated. One very common way to do this is via the method of finite differences [9]. In finite differences the gradient is estimated by perturbing the operating point along each dimension individually. The equation for the k^{th} operating point, for a D dimensional problem is

$$g_k(x_k) = \begin{pmatrix} \frac{f(x_k+\delta_k\xi_1)-f(x_k-\delta_k\xi_1)}{2\delta_k} \\ ... \\ \frac{f(x_k+\delta_k\xi_D)-f(x_k-\delta_k\xi_D)}{2\delta_k} \end{pmatrix},$$

where the parameter δ_k denotes the resolution of the gradient estimate; the limit as $\delta_k \to 0$ is the instantaneous gradient, and the vector ξ_d is a vector with a one in the d^{th} dimension. This estimate takes $(2D)$ function evaluations for each gradient estimation.

Simultaneous perturbation stochastic approximation (SPSA) stochastically chooses a single direction in which to perturb the operating point to estimate a gradient. This improves upon the finite difference method by reducing the number of objective function evaluations required to a constant (two), regardless of the dimensionality [52]. The gradient estimation for SPSA is

$$g_k(x_k) = \frac{f(x_k + \delta_k \Delta_k) - f(x_k - \delta_k \Delta_k)}{2\delta_k} [\Delta_{k,1}^{-1}, \Delta_{k,2}^{-1}, ..., \Delta_{k,D}^{-1}]^T.$$

The vector Δ_k contains random values and replaces the series of ξ_d vectors. The randomness of the Δ_k vector components adds a stochastic element to the gradient estimations, but does not add bias. It has been suggested that this stochastic element results in a more robust algorithm [32].

Many gradient estimates do not provide the exact gradient, or instantaneous slope at a single point. Instead, they represent a trend over a given region or resolution. In SPSA, for example, this resolution is $2 \times \delta$. This is advantageous because when the functional space has local minima or noise, the exact gradient may not be that useful. To avoid local minima or ignore noise, one may be interested in estimating what direction constitutes "downhill" over a region. Thus, we will refer to the gradient over a region S_p as the vector β^* that best fits a hyperplane to the function over the region S_p, that is:

$$[\beta^*, \beta_0^*] = \underset{\beta, \beta_0}{\operatorname{argmin}} \int_{x \in S_p} (f(x) - \beta^T x - \beta_0)^2 dx. \tag{1.2}$$

Many of the algorithms that fall into the category of gradient descent algorithms actually use an approximation of this type of regional trend instead of an instantaneous gradient. MEGA makes this explicit by estimating gradients at various resolutions to guide its search. Chapter 3 has a detailed discussion of the estimation of these regional gradients.

Another algorithm that uses this type of regional trend information is the particle swarm optimization (PSO) algorithm. Particle swarm is an evolutionary algorithm, in which many agents, or particles, search the space [14]. The agents' behavior is modelled after the natural

flocking behavior of birds, giving each agent a velocity to move it towards nearby promising areas. On the i^{th} iteration each particle p is updated independently using the equations:

$$\overline{v}_i = c_0 \times \overline{v}_{i-1} + c_1 \times \rho \times (\overline{pb} - \overline{p}_{i-1}) + c_2 \times \rho \times (\overline{gb} - \overline{p}_{i-1}), \quad (1.3)$$

$$\overline{p}_i = \overline{p}_{i-1} + \overline{v}_i, \quad (1.4)$$

In these equations $\overline{p}_i$ is the position of particle p for the i^{th} iteration, $\overline{pb}$ is the current personal best for that particle, $\overline{gb}$ is the current global best for all the particles, and ρ is a random number between zero and one. The equations also require weighting factors, c_0, c_1 and c_2. Setting these parameters can be challenging, and there are a number of recommendations for their values. The number of particles used is also an adjustable parameter. There is a summary of current PSO information at [1]. In the PSO implementation each agent moves with a velocity that is towards both a personal best position and the global best position. The personal best is set for each agent, and is the location, searched by that agent, with the best objective function value. The global best is the location with the best objective function value overall. The two best values are updated each iteration. The PSO velocity may be viewed as a type of gradient in that it points towards known good regions. The weighting factors mean that the velocity searches in a promising direction but does not precisely stop in the personal or global best locations. This is an important factor because it helps the search avoid getting trapped in local minima.

One slightly different method for estimating the regional gradient is to estimate the functional shape and then calculate the gradient. This method is represented in the class of response surface methods, in which the functional shape is estimated and the gradient determined for the estimated function [26].

1.2.2 Multiresolution Search

One of the challenges in global optimization is doing both a search on a global scale to locate the region of the optimum, and then a search on the local scale to search the region of attraction for that global optimum. The term multiresolution search refers to a simultaneous search on multiple scales. One way to do this is to use a hybrid algorithm combining some

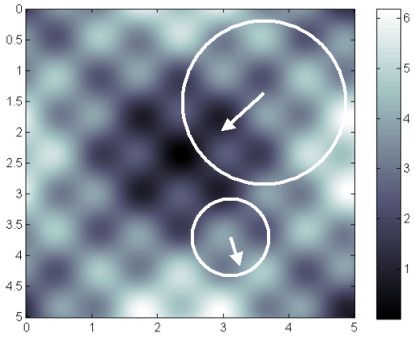

Figure 1.2: This figure shows an objective function with two resolutional gradients estimated with a difference method. One gradient is at a small resolution, and one is at a large resolution. This figure shows how different the two estimates are, as well as depicts how the larger resolution may help avoid local minima.

form of stochastic search for the global portion and a more structured gradient descent type algorithm for the last step. Some details of hybrid algorithms can be found in [63, 45].

Multiresolution search can be incorporated into a seamless algorithm if there is a mechanism for search both locally and globally. Tabu search describes this process explicitly with the processes of 'intensification' (combining known good solutions to explore their local region), and 'diversification' (searching previously unexplored areas). Intensification serves to search the local space, while diversification is a global search. Another way to do this is to allow for steps to occur on different scales, such as by dividing the search space into different neighborhoods based on areas of attraction for local minima. Locatelli provides a useful description of the motivation for this approach [30].

In other cases the multiresolutional nature of the search is less explicit. For example, some simulated annealing implementations use cooling schedules which allow initial random searching on a global scale, but over time limit the searching to a more local level [52]. An annealing approach that allows many hill-climbing steps in the beginning helps the global aspect of the search. PSO, as noted in the previous section, implements the idea implicitly by estimating a regional gradient using both the local best and global best operating points.

In this research, the algorithm is specifically designed for searching both globally and locally. As described in Chapter 2, the MEGA global optimization method is multiresolutional as regional gradients are estimated at different resolutions, and steps are taken at each resolution.

1.2.3 Memory

In optimization algorithms memory is used to keep track of which areas are promising, which areas are known to perform poorly, and which areas have not yet been searched. As mentioned previously, the Tabu search explicitly lists areas that have been searched and how they performed.

More commonly, algorithms implicitly use memory by basing future steps on previous operating points. In SPSA, for example, the gradient is estimated based on the current location, and a step is taken based on that location. This effectively uses a memory of one point in directing the future search. In PSO the new velocity is based on the behavior in the previous iteration, plus a record of the best points found by the particle and in the overall search. In this way, the effective memory includes a component summarizing the search up to that step.

Another way of looking at memory is to base the attractiveness of a solution component on the performance of the solutions containing that component. This is the approach taken by ant system optimization, which is based on the behavior of real ants [13]. The algorithm uses many ants (agents), and has similarities to other population-based algorithms (genetic algorithms (GA), PSO, MEGA). The ant system is designed to solve combinatorial optimization problems, such as the travelling salesmen problem. In this formulation each segment of the path is given an attractiveness (a pheromone marking) inversely related to the length of the total tour. Segments that are used in shorter paths will have a stronger pheromone marking, and individual agents will choose to use a segment with a probability related to the strength of the pheromone marking. This system means that good solutions are reinforced, in a type of positive feedback loop. Similar positive feedback is used in GA where genes that are a part of a highly performing solution are more likely to be passed

on to the next population. Positive feedback is seen in a less explicit way in MEGA where highly performing points are kept in the database and poorly performing points are not.

This chapter has detailed a wide variety of existing optimization techniques and algorithms. It is clear that there are many different search strategies that can be advantageous. In the research work associated with this thesis the driving motivation was to use all available function information to increase the efficiency and effectiveness of global optimization. To do this the algorithm reuses previous evaluations of the objective function, possibly many times. This approach is not unique to the MEGA algorithm; Custódio et al. use a similar motivation to modify a pattern search algorithm [8]. However, the MEGA algorithm is unique because it uses available information through the three identified techniques of gradient descent, multiresolution search, and memory. The following chapters detail not only the novel MEGA algorithm, but also look at how an exploration of these three techniques may be used to improve existing algorithms.

Chapter 2

THE MEGA ALGORITHM

The multiresolution estimated gradient architecture (MEGA) algorithm is designed to solve problems with some structure, as one might expect to see in engineering applications. This algorithm uses estimated regional gradients to globally search the space. The development and exploration of the MEGA algorithm comprises a significant portion of this thesis work. In this chapter the MEGA algorithm is described completely in section 2.1. A series of tests is presented and strengths and weakness of the algorithm are discussed in section 2.2. Section 2.3 looks at some of the performance properties of the MEGA algorithm. In particular, consideration is given to the problem features that would allow MEGA to be used successfully.

Based on the initial performance of the algorithm a series of improvements were conceived and tested. Proposed variations of MEGA are aimed at making the algorithm both more effective, and more efficient. These modifications are discussed in section 2.4. This section leads into Chapter 3 wherein proposed improvements to the gradient estimation techniques used in MEGA are explored in depth.

2.1 Algorithm Description

The MEGA algorithm is an iterative scheme where multiple new operating points are determined and evaluated in each iteration. Within the algorithm there are a few sub-processes that can be implemented in a variety of ways. The general structure for solving a D dimensional search problem is presented below in pseudo-code, then following sections discuss the details of the different sub-processes.

Sample initial points $\{x_i\}$ for $i = 1$ to $(D+1)^2$

Evaluate initial points $\{f(x_i)\}$

$x^* \leftarrow \text{argmin}_{\{x_i, i=1...(D+1)^2\}} f(x_i)$

Build database with $\{x_i, f(x_i)\}$

while convergence criteria not met **do**

  cluster the set of points $\{x_i\}$ into $(D+1)$ local neighborhoods

  for all n=1 to $(D+1)$ **do**

    Estimate local regional gradient β_n^* based on $\{x_i, f(x_i)\}$ in neighborhood n

    Calculate new operating point x_n in direction β_n^*, located σ from the cluster centroid

    Evaluate new point $f(x_n)$

    Update database with $(x_n, f(x_n))$

    if $f(x_n) \leq f(x^*)$ **then**

      $(x^*, f(x^*)) \leftarrow (x_n, f(x_n))$

    end if

  end for

  Estimate global regional gradient β_g^* based on new operating point pairs $\{x_n, f(x_n)\}$

  Calculate new operating point x_g in direction β_g^*, located σ from the cluster centroid

  Evaluate new point $f(x_g)$

  Update database with $(x_g, f(x_g))$

  if $f(x_g) \leq f(x^*)$ **then**

    $(x^*, f(x^*)) \leftarrow (x_g, f(x_g))$

  end if

end while

2.1.1 Initialization

To initialize the MEGA algorithm a set of starting points $\{x_i\}$, $i = 1, \ldots, (D+1)^2$ are chosen randomly from a uniform distribution over the search space S. The objective function is evaluated at each of the starting points to form the initial sample pairs $\{x_i, f(x_i)\}$. If knowledge of the search space allows, prior guesses of the optimal predictor can be used instead of random samples for the initial set. Similarly initial points can be randomly generated using a non-uniform distribution over the search space. If the prior guesses are

good this may improve the architecture of the algorithm, but poor guesses may hinder the search. The default behavior, therefore, is to use the uniform distribution.

The number of points needed for the MEGA algorithm is dictated by the number of points needed to estimate the gradients. Linear regression is used for each gradient estimate, thus for a D dimensional objective function, a minimum of $D+1$ sample points are needed to create a numerically stable estimate. Then, the new points generated by each local step are used to estimate a global gradient; a minimum of $D+1$ local estimates are required for a full rank estimate for the global gradient. This results in a total of $(D+1)^2$ initial points. Thus, the number of initial points scales with the square of the dimensionality of the problem, while the number of parallel searches scales linearly with the dimensionality.

2.1.2 Clustering

At each iteration of the algorithm the database is re-clustered into $D+1$ neighborhoods. An agglomerative (bottom-up) average-linkage clustering of all the database samples separates the samples into $D+1$ mutually exclusive clusters. Initially all samples are considered clusters. Then there are $(D+1)^2-(D+1)$ iterations of the clustering algorithm, and at each iteration the two clusters that are closest are joined. Using an average-linkage implementation, the distance between two clusters is defined as the average distance between elements in each cluster. K-means and other clustering methods were also tried, but preliminary results suggested that there was little gain in the optimization for the larger computational costs.

A number of global optimization techniques use an initial clustering of points to begin local searches or to define sub-domains for search [51]. In MEGA, the clustering happens every iteration and is a way to incorporate local memory into the gradient estimation for the local gradient descent steps. The MEGA clustering is a computationally-intensive step, particularly as the dimensionality increases. In preliminary experiments, only re-clustering every q iterations for varying values of q was tried. The results significantly deteriorated. This suggests that the clusters are intermixing between iterations, and that this intermixing is significant for good performance.

2.1.3 Gradient Estimation

The gradients over the regions defined by each neighborhood of previously evaluated points are estimated using standard linear regression to fit a hyperplane $f(x) = \beta^T x + \beta_0$ to the sample pairs $\{x_i, f(x_i)\}$ by minimizing the total squared error. For example, let the nth neighborhood contain the sample pairs $\{x_i, f(x_i)\}$ if $i \in \mathcal{J}_n$. Then the estimated regional gradient is β^*, where

$$[\beta^*, \beta_0^*] = \underset{\beta,\beta_0}{\operatorname{argmin}} \sum_{i \in \mathcal{J}_n} (f(x_i) - (\beta^T x_i + \beta_0))^2, \tag{2.1}$$

where β_0^* is an offset that does not form part of the regional direction.

The nth local gradient estimate is formed by fitting a hyperplane to only the sample pairs $\{x_i, f(x_i)\}$ in the nth neighborhood. The global gradient estimate is formed by fitting a hyperplane to the $D + 1$ newly estimated local sample points $\{x_n, f(x_n)\}$.

Unlike some gradient-based optimization methods, such as finite-differences or SPSA, no new points are evaluated in order to estimate the gradient. Using previously evaluated points to do a least-squares hyperplane fit is an approach also used in first-order response surface methods [37].

We define a regional gradient to be the average gradient over a region of the operating space, as described above. The regional gradient is explored in depth in Chapter 3. In the MEGA algorithm this estimated gradient is used to direct the search. Traditionally x_{new} is calculated using the equation $x_{new} = x_{old} - \sigma\beta^*$, where σ is a stepsize parameter. In MEGA, the point from which to move, x_{old}, is not pre-defined, because the gradient represents a region instead of a value at a single point. Instead, the step is taken from the centroid of the region represented by the gradient. That is,

$$x_{new} = \frac{\sum_{i \in \mathcal{J}_n} x_i}{\|\mathcal{J}_n\|} - \sigma\beta_n^*,$$

where $\|\mathcal{J}_n\|$ denotes the cardinality of the set $\mathcal{J}_n$.

The least-squares regression coefficient vector β^* has a closed form solution:

$$[\beta^* \beta_0^*]^T = (X^T X)^{-1} X^T Y \tag{2.2}$$

where X is a matrix with the sample point x_i in the ith row, and Y is a vector with $f(x_i)$ in the ith row, and the $i+1$th row of X and Y is all ones. The $(D+1) \times (D+1)$ matrix $X^T X$ must be invertible in order to form the gradient estimate. Thus, $D+1$ linearly independent sample points are needed. Before calculating each gradient estimate, a check is made to determine if there are enough linearly independent sample points. If not, another point from the database is randomly chosen and added to the estimation. Randomly chosen points from the database are added one-by-one until sufficient linearly independent samples are available for a numerically robust matrix inversion.

2.1.4 Database Management

MEGA maintains a database of previously-evaluated points which serves as the memory for the algorithm. Not all previously-evaluated points are kept in memory. At any given time there are $(D+1)^2$ points are in the database, one of which is noted to be the current best operating point: x^*. The current best operating point is replaced by some other point $\tilde{x}$ if $f(\tilde{x})$ is evaluated and $f(\tilde{x}) < f(x^*)$.

The database is initialized at the start of the optimization with $(D+1)^2$ randomly drawn pairs $\{x_i, f(x_i)\}$. Each time a new operating point $(x_{new}, f(x_{new}))$ is evaluated the database is updated to include that point. When a new point is added to the database, some point is removed. In this way the size of the database is maintained. Every time the nth neighborhood adds a point, the worst point (before the new addition) in the nth neighborhood cluster is removed. Every time the new global point is added, the worst point in the entire database (before the new addition) is removed. A separate list of these inactive points may be maintained so that the algorithm can draw on them if needed.

The requirement that gradient estimates be based on $D+1$ linearly independent samples, and the frequent re-clustering of points combine to limit the likelihood of spending infinite iterations on a local search around a relatively poor local optimum. Replacing the worst point with a new point in each iteration reduces the likelihood that the same points will be used in each search step, and therefore that the search will stagnate.

At all times, the database contains $(D+1)^2$ active points, $D+2$ of which are replaced

each iteration, allowing memory to persist over many iterations. Each new point x_{new} is added regardless of $f(x_{new})$. If x_{new} is a relatively poor performer, then this information will inform the next estimated gradient to direct the search away from x_{new}.

2.1.5 Parameter Settings

Most optimization algorithms have some parameters that can be tuned to improve the algorithm's performance on a specific problem. In the MEGA design an effort was made to minimize any free parameters, and to provide defensible reasons for any parameter choices. Ideally, an algorithm would automatically and optimally adjust its own parameters to a new objective function.

The first MEGA parameter is the number of random sample points for the initialization. We choose the minimum number of points required, $(D+1)^2$, as detailed above in the initialization section. Thus, this parameter is coupled to the decision to use $D+1$ local searches. The number of local searches was chosen to provide the minimum number of local new points to create a full rank least-squares estimate for the global search. It should be noted that this results in a number of active points that is exponentially related to the dimensionality of the problem. The authors hypothesize that having the number of active points directly related to the dimensionality of a problem is advantageous for fully searching the space, but a linear relationship may work as well as an exponential one.

Another set of parameters is the starting value and decay schedule for the stepsize σ. If the stepsize is too large, new points will be located seemingly at random. If σ is too small, the new points will not be significantly different from existing points. The starting σ is set to be half the length of the longest edge of the hyperrectangular space. This start value ensures that early iterations of the algorithm form a coarse painting of the space: any given step can move entirely from one quadrant of the search area to another one. The decay schedule is a monotonic geometric decay schedule,

$$\sigma_{j+1} = \sigma_j \times \varepsilon,$$

where j indexes the iteration of the MEGA algorithm. The algorithm seemed robust to a range of possible values for ε, which was set to a 0.9 after trial runs with a variety of values.

2.1.6 Boundary Conditions

The MEGA algorithm has been designed to work with black-box type objective functions. The search space for these functions is usually constrained, as the variables have practical ranges. In this chapter it has been assumed that all constraints are pre-determined limits on individual dimensions, so that the search space S is a hyper-rectangle (other types of constraints are discussed in Chapter 4). Hyper-rectangles have deep pockets in high-dimensional spaces, or equivalently, much of the volume of the space is in the corners. This can be understood by considering a hypersphere of radius of one centered inside a hypercube of edge length two. The volume of the hypersphere decays to zero as the dimension increases. The volume of the hypercube grows without limit. The hypersphere touches the edges of the hypercube for any given dimension. Thus, the volume of the search space is predominantly in the corners of the hypercube as the dimension grows. Related to this property, uniformly random sample points are likely to be near the edge of the search space. This means that optimization algorithms searching the space by steepest descent are likely to have many steps that land outside the search space. If the step size is small enough to keep the search points inside the search space the search may progress slowly. (For more details on the curse of dimensionality see [19].)

In early incarnations of this work the constraints were enforced by limiting the values the search variables could take on, and vector values violating the constraints were mapped to the closest boundary point along the estimated gradient. However, this method could cause sample points to cluster at the edge of a boundary. A cluster of such points would not be linearly independent, leading to difficulties in estimating gradients. This results in redundant searching in the best of cases.

For this reason, the boundaries are handled using slack variables [42], in which there is a penalty applied for new points that fall beyond the boundaries of the search space. If a new operating point x falls outside the limits of the search space S, MEGA gives x the value

$$\hat{f}(x) = f(x) + \sum_{d=1}^{D} e^{\|\delta_d\|}, \tag{2.3}$$

where δ_d is the distance x is outside the search space in the dth dimension.

2.2 Test Suite and Results

Testing of optimization algorithms is quite difficult. This is because there are many conflicting ways to evaluate algorithms, and also because it is difficult to compare an algorithm to a good sample of competing algorithms. Some discussion of algorithm evaluation may be found in Chapter 1. For this work the MEGA algorithm was initially tested on a number of different test function adapted from those found in the literature, including [62, 2]. This testing looked at the convergence rates and number of function evaluations required as the number of dimensions in the search space increased. The results showed promising performance statistics.

The analysis presented in this thesis compares the MEGA algorithm to published results. The first set of comparisons is to the PSO algorithm. This algorithm is similar to MEGA because it uses a velocity which is an estimate of the regional gradient to direct the search, and also includes both local and global information. A more complete discussion of these similarities may be found in section 1.2. The second set of comparisons is to direct search algorithms. These approaches differ from the MEGA approach in that they use very little calculation to determine promising search areas. This makes them extremely robust because they don't rely on functional shape or information. Direct search algorithms are popular as well because they are easy to use and lend themselves to statistical analysis.

Five standard test functions, four of which are seen in Figure 2.1, were chosen to stress different aspects of an optimizer. The standard Rosenbrock function is strictly convex, but considered challenging for algorithms using steepest descent because it has a narrow valley. The sinusoidal function is non-convex, and has only a slight global trend towards an embedded minimum, but the highly structured space lends itself to solution via MEGA's gradient search. The Griewank function is bowl-shaped but with many local minima within the search space. While we anticipate that MEGA will be able to capitalize on the bowl-shape, the Griewank function still poses great difficulty because of the large number of local minima. The Branin function is a two dimensional function which poses a challenge because there are three non-embedded global minima. Similarly, the Langerman function is largely unstructured, and it is computed on a ten dimensional search space. We anticipate that the

direct search methods examined in the second segment of the testing will do comparatively well with the Branin and Langerman function because they do not rely on structured spaces.

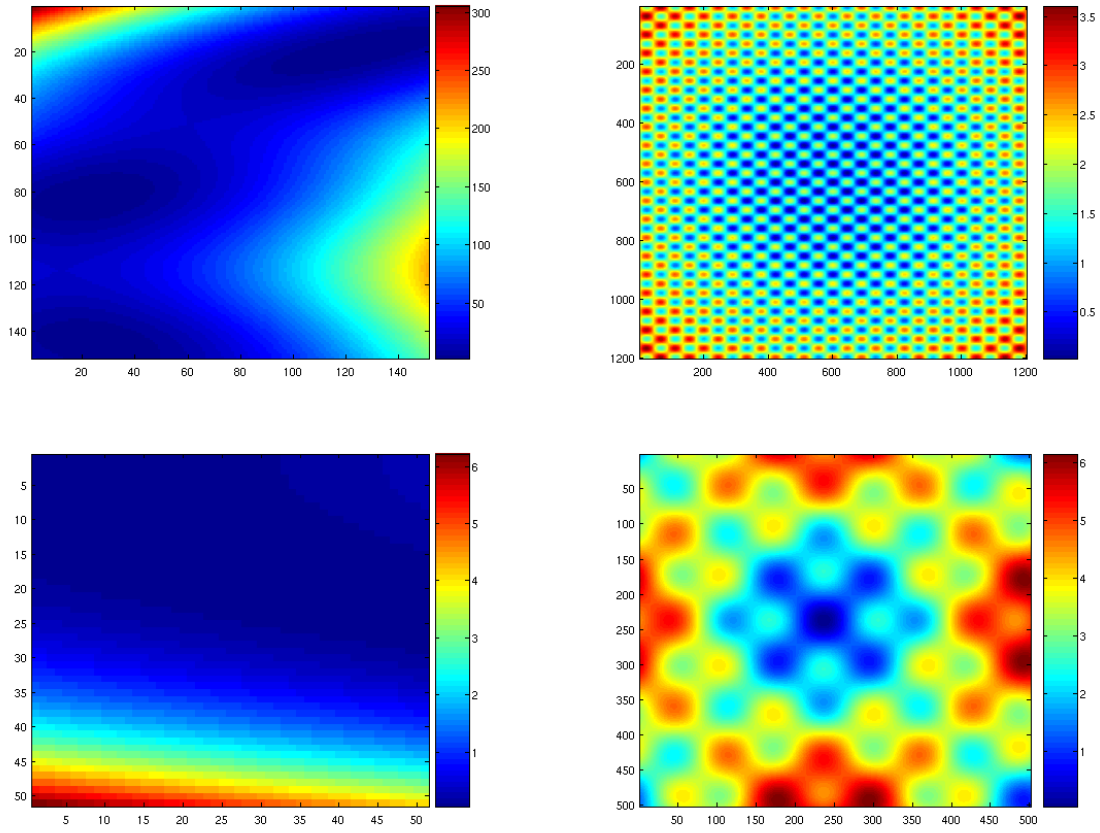

Figure 2.1: Two-dimensional instances of four test functions. Clockwise from upper left: Branin, Griewank, sinusoidal, Rosenbrock. The ten-dimensional Langerman function is not shown.

Recently, Schutte and Groenwold compared a number of PSO variants [49]. MEGA is compared to one of the two variants they recommend based on their study, namely PSO with constriction (CPSO). The update equations for CPSO differ only slightly from (1.3) and include an added parameter χ which constricts the velocity.

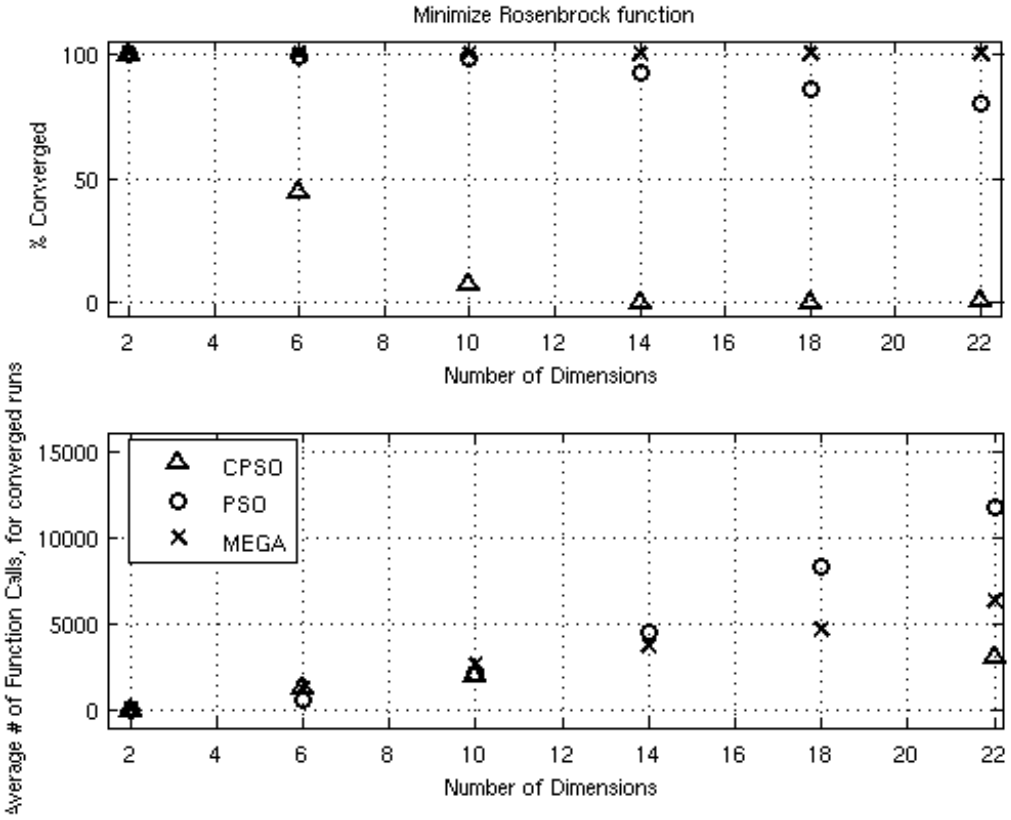

Figure 2.2: Statistics for minimizing the Rosenbrock function. Top: percentage of the 100 test runs that converged. Bottom: average number of function evaluations for the converged runs. (Runs that did not converge are not included in this average.)

$$\begin{aligned} \overline{v}_i &= \chi(c_0 \times \overline{v}_{i-1} + c_1 \times \rho \times (\overline{pb} - \overline{p}_{i-1}) + c_2 \times \rho \times (\overline{gb} - \overline{p}_{i-1})), \\ \overline{p}_i &= \overline{p}_{i-1} + \overline{v}_i, \end{aligned}$$

The algorithm was implemented using the parameters recommended by Schutte and Groenwold [49]. Test results are also shown for a standard PSO implementation in which the velocity weight parameters are equivalent to those used in CPSO, essentially $[c_0, c_1, c_2]_{PSO} = \chi[c_0, c_1, c_2]_{CPSO}$. The number of agents differs between the two implementations, as CPSO

uses 25 agents for each test run, while the alternate PSO implementation uses $(D+1)^2$ agents. It should be noted that this is equal to the number of active sample points in any given iteration of the MEGA algorithm.

Here, as in Schutte's tests [49], a method is said to have *converged* for a function $f(x)$ if the search has discovered some $\bar{x}$ such that $f(\bar{x}) - f(x^*) \leq \epsilon$, where x^* is defined as in (1.1), and $\epsilon = .001$. Similarly, each algorithm was allowed to run for a maximum of $1000D$ function evaluations. The results are examined in terms of percent convergence and average number of function evaluations (given only the converged instances). These values are plotted as a function of dimension for each of the test functions.

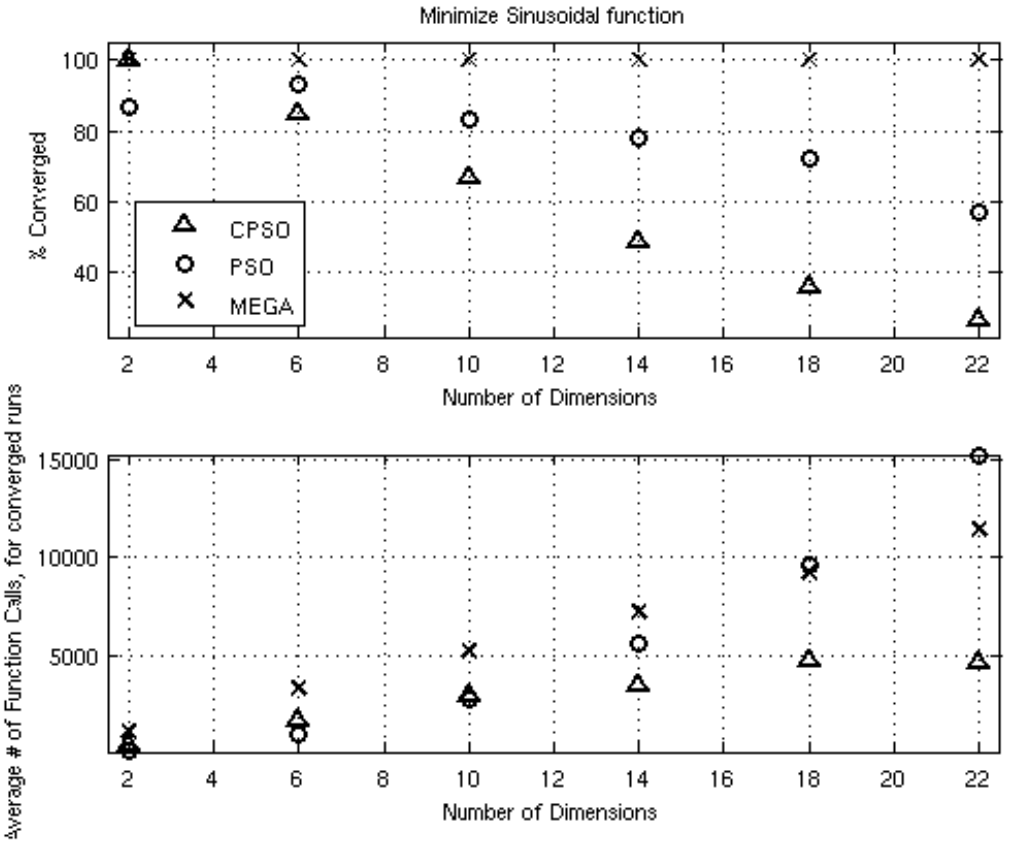

Figure 2.3: Statistics for minimizing the sinusoidal function. Top: percentage of the 100 test runs that converged. Bottom: average number of function evaluations for the converged runs. (Runs that did not converge are not included in this average.)

The Rosenbrock function is one of the best known test functions. Its long valley makes it particularly hard for algorithms to find the precise minimal location. The results of running all three test algorithms on this function are shown in Figure 2.2. The results for the sinusoidal test function are shown in Figure 2.3, and for the Griewank function in Figure 2.4. Of the test functions in this section, the Griewank function posed the most difficulty, most likely due to the high number of local minima within the search space. There are more results for the Griewank function presented in the second part of the testing.

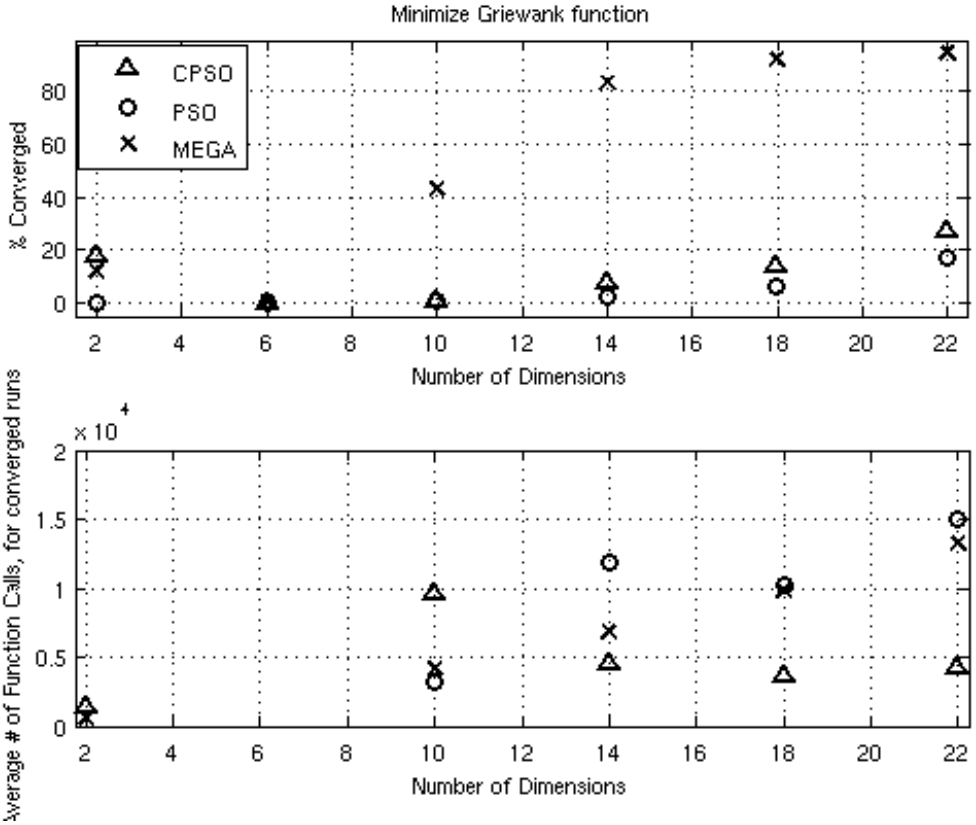

Figure 2.4: Statistics for minimizing the Griewank function. Top: percentage of the 100 test runs that converged. Bottom: average number of function evaluations for the converged runs. (Runs that did not converge are not included in this average.)

Examining the plots for three test functions we see that all three algorithms perform

similarly in low-dimensional spaces, converging frequently with relatively few function evaluations. An interesting aspect of the MEGA design is that it specifically scales with dimension, and converges consistently even when the dimensionality increases. The performance of the other two algorithms, however, drops off as the number of dimensions is increased. The Rosenbrock is easily solved by the PSO and MEGA algorithms, but the CPSO algorithm has difficulty at higher dimensions. The Griewank function is an interesting case because all the algorithms have some difficulty converging at lower dimensions. This is the only case in which MEGA is outperformed by CPSO in the two-dimensional search. The MEGA algorithm is able to successfully find the global minimum more frequently as the number of dimensions increases.

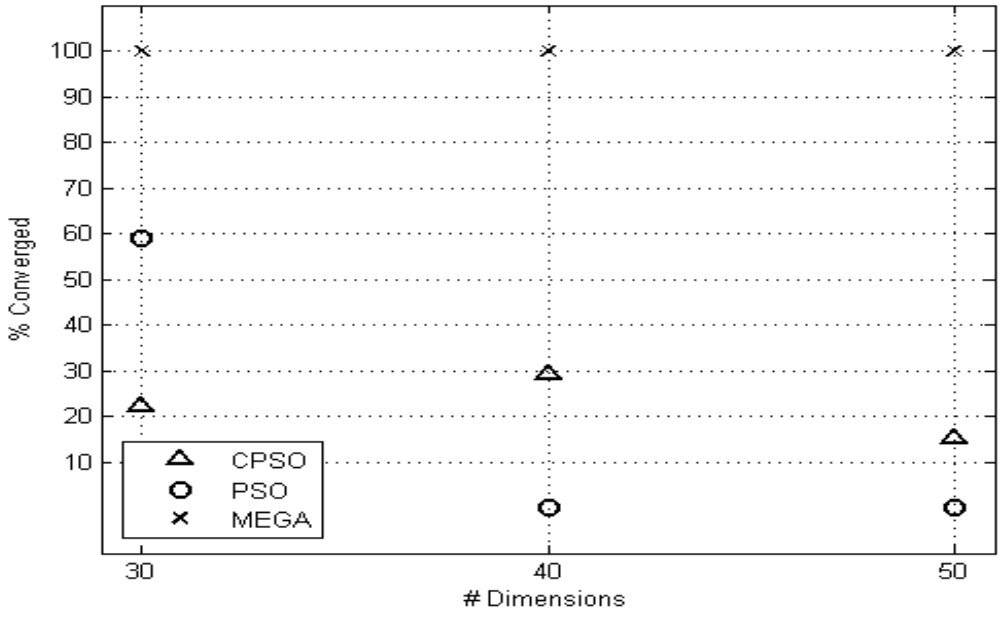

Figure 2.5: Percentage of the 100 test runs that converged for finding the optimum of the sinusoidal function over higher dimensions.

It is also interesting to look at the number of function evaluations needed for convergence. MEGA requires fewer function evaluations for the Rosenbrock function, presumably because this convex space is well suited to gradient direction search. MEGA also required fewer

function evaluations than its competitors as the number of dimensions is increased. In some cases the PSO variants required fewer function evaluations at low dimensions.

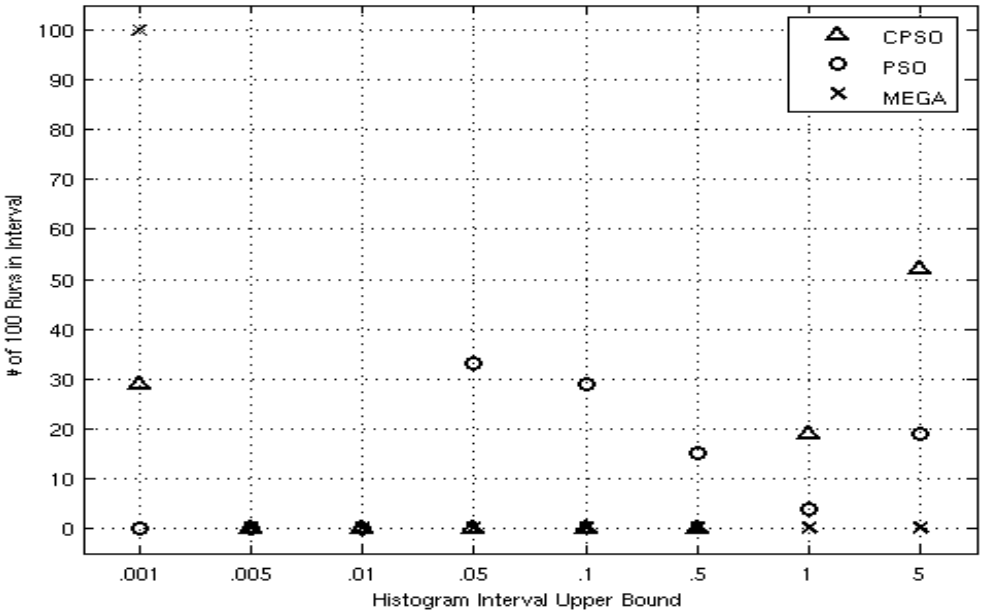

Figure 2.6: Histogram of the final values of the 100 runs with the sinusoidal test function for 40 dimensions.

While this testing showed that MEGA offers consistent performance at relatively low dimensionality, further testing was performed to determine if MEGA could maintain its consistent convergence rates at even higher dimensions. Figure 2.5 compares the performance for optimizing the sinusoidal function over 30, 40, and 50 dimensions, where the maximum allowed function evaluations was again $1000D$. MEGA converged 100% of the time for 100 runs in each dimension. The results are a function of the maximum number of function evaluations allowed and the convergence criteria threshold ϵ. A more complete picture is developed by considering Figures 2.6 and 2.7, which show histograms of the final values of the 100 runs for 40 dimensions and 50 dimensions. These figures show that the results would not vary with moderate changes in ϵ. The number of PSO and CPSO runs that ended at quite high function values suggests that it would take significantly more function

evaluations to converge to the precise optimum, if the algorithms were not trapped in local optima. These results are all for the sinusoidal test function.

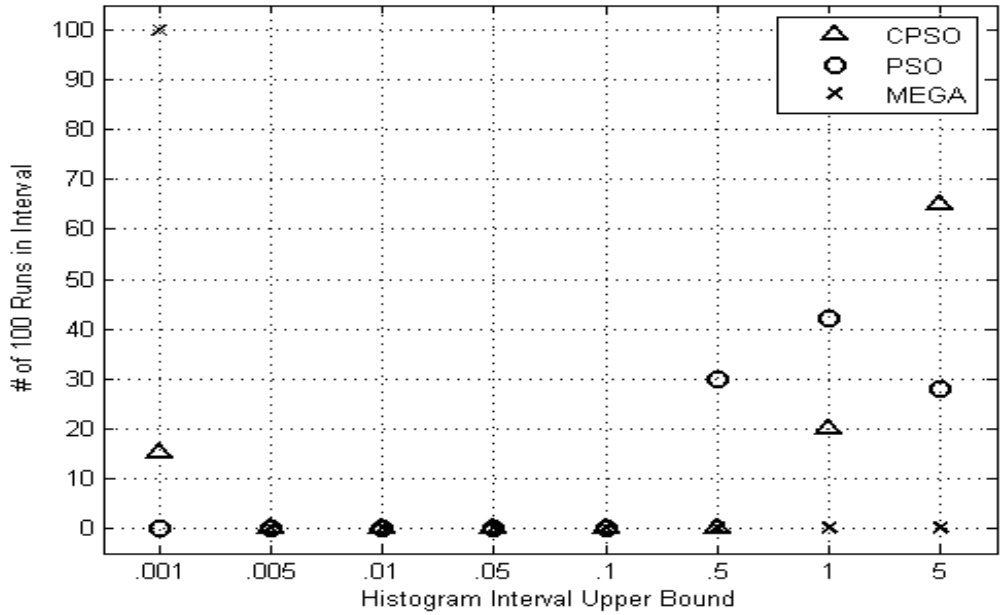

Figure 2.7: Histogram of the final values of the 100 runs with the sinusoidal test function for 50 dimensions.

As an additional note, we can use these histograms to identify differences between the performance of the PSO and CPSO algorithms. As the number of dimensions increases the PSO algorithm converges with greater reliability, most likely because of the increased number of agents. When the CPSO algorithm does converge it generally takes fewer function evaluations, which is consistent with the discussion of population size in the reference paper. It is worth noting that the performance of the CPSO algorithm was qualitatively compared to that presented in the reference paper (when possible) to ensure that the implementation was correct.

The previous section shows that the MEGA algorithm performs competitively when compared to a similar algorithm. The specific comparisons look at the convergence rates of the algorithms, and how the algorithms scale with dimensionality. The next results

compare the MEGA algorithm to existing results for direct search algorithms, and focus on how the function values change as the number of function evaluations increases. In [2] data are available for simulated annealing type algorithms, improving hit-and-run (IHR) and hide-and-seek (HNS), as well as population based algorithms controlled random search (CRS), genetic algorithms (GA), and differential evolution (DE). Additional figures provide comparisons so that MEGA could be evaluated against these results.

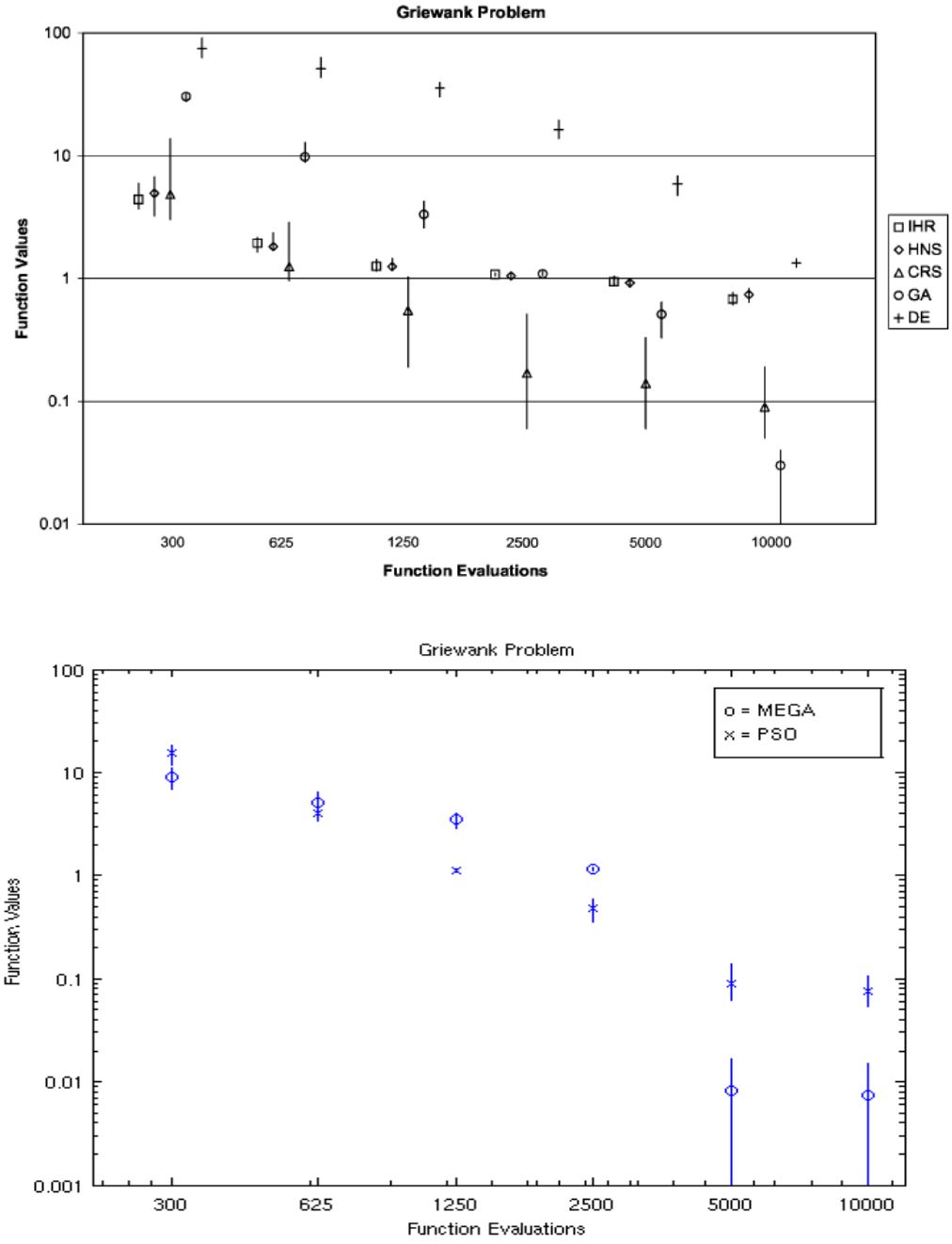

Figure 2.8: Results for the ten dimensional Griewank problem. Results from [2] are on the top, and from this research on the bottom. It should be noted that the Y axis scaling is expanded for the MEGA results, because the final values fall lower than would be shown with the scaling for the Ali et al. results.

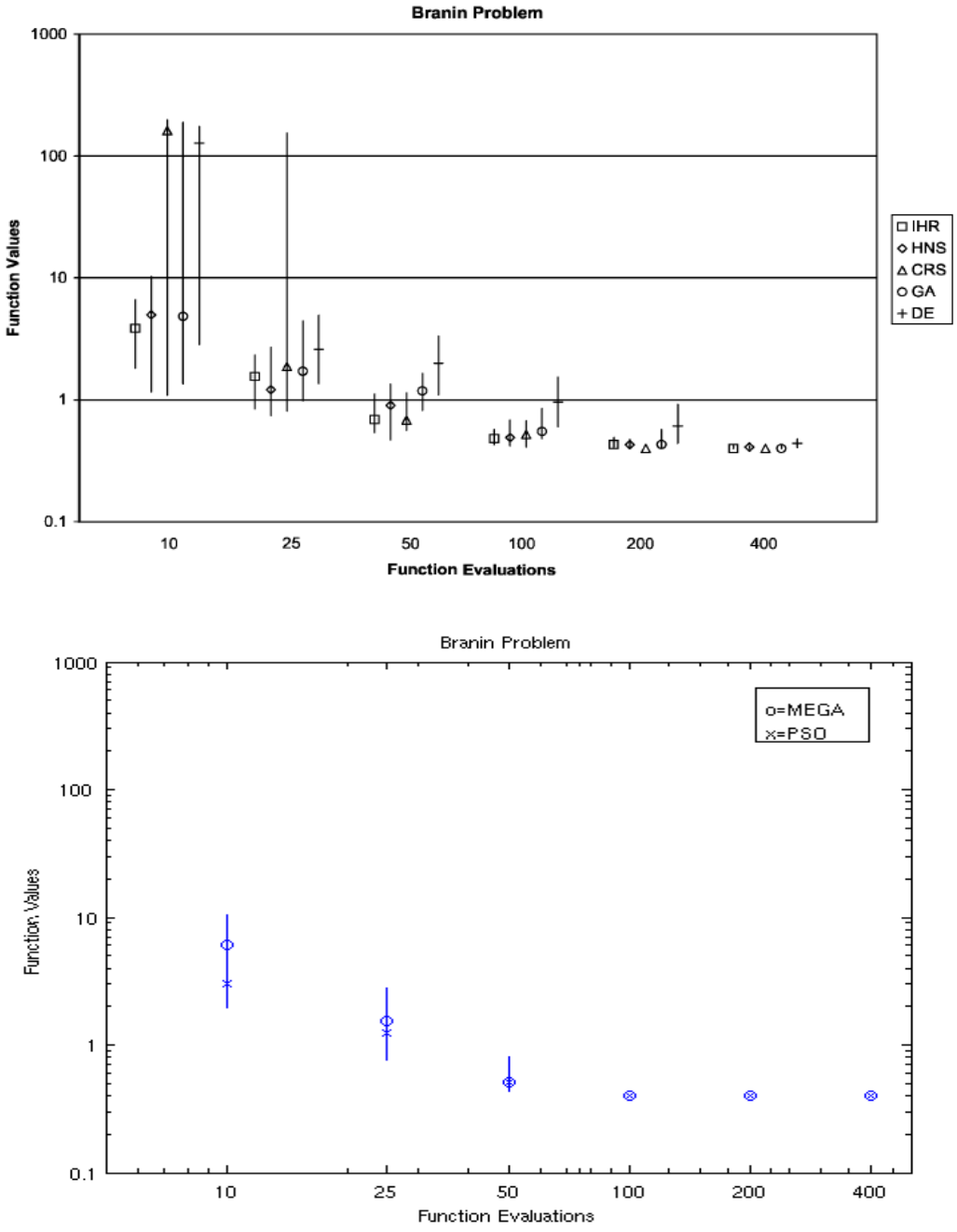

Figure 2.9: Results for the two dimensional Branin problem. Each algorithm attained the minimum value for the Branin problem, of $\frac{5}{4\pi}$.

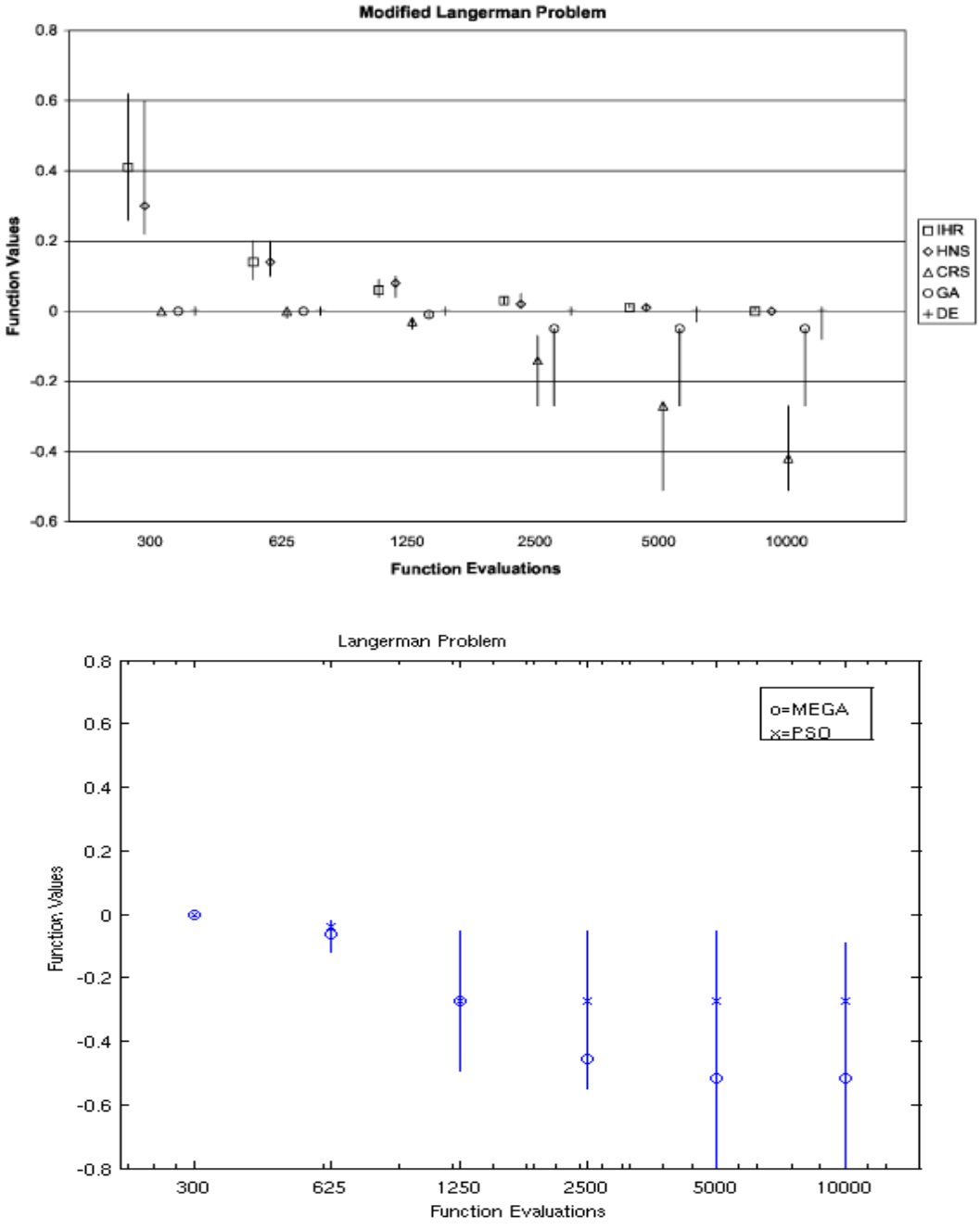

Figure 2.10: Results for the ten dimensional Langerman problem. The global minimum, at -0.965, proved to be difficult to find. It should be noted that the Y axis scaling is expanded for the MEGA results, because the final values fall lower than would be shown with the scaling for the Ali et al. results.

Looking at the Griewank results, in Figure 2.8, we can see the the MEGA algorithm is competitive with the best performers from the direct search methods. At early function evaluations MEGA is finding function evaluations as low as the best of the algorithms tested in the original paper. The CRS algorithm outperforms the MEGA algorithm for up to about 2500 function evaluations. Beyond that, however, MEGA continues to improve on its solutions, ultimately improving on the direct search algorithms by an order of magnitude. The performance results for PSO, with parameters as described above, are also included in this figure. From about 625 to 1250 function evaluations the PSO algorithm finds a smaller objective function value on average. However, at the end of the runs MEGA finds objective function values much smaller. These results suggest that early on the direct search algorithms are able to find promising search locations, but that the MEGA algorithm does a better job finding the global minimum from those areas of attraction.

Figure 2.9 presents the results for testing on the Branin problem. The Branin problem is not as structured so it poses a larger challenge for the MEGA algorithm. The performance across all the algorithms is similar, particularly as the number of function evaluations increases. The MEGA algorithm performs similarly to the best of the direct search methods. After 100 function evaluations both PSO and MEGA are converging in all of the test runs. This suggests that the MEGA algorithm is as good a choice as any of the direct search methods, but not necessarily a better choice.

Testing was also done on the Langerman problem, which is a non-structured problem in ten dimensions. The area of attraction for the global solution is small relative to the size of the search space, so this is a challenging problem. The direct search methods have an advantage because they do not rely on finding structure in the problem. However, MEGA performed surprising well, having better average results than the other algorithms tested. Also, while PSO did relatively well on the two dimensional Branin problem, it does not work as well on the ten dimensional Langerman problem. Both PSO and MEGA have a large variance on the function values for high numbers of function evaluations. This implies that, in a significant number of runs, the algorithms are getting trapped in a local minimum. This is similar to the results from the direct search algorithm which often got trapped in a local minimum of the same function value.

The second set of results presented in this section give interesting insight into the performance qualities of the MEGA algorithm. A number of the other algorithms are finding lower values during the early function evaluations, but if the algorithms are allowed to run for a while, MEGA converges more reliably to the global minimum value. If the goal is to obtain a somewhat good performance value in a very few objective function evaluations then MEGA is not the best choice of algorithms. If it is necessary to find the absolutely lowest objective function value, and more time is affordable, then MEGA is a better choice. MEGA may also be more desirable if the dimensionality of the search space is high. The goal of future work is to improve the MEGA algorithm so that it outperforms the competitors on all types of performance measures.

2.3 MEGA Performance Properties

Of interest in any search algorithm are the convergence properties associated with it. In particular, can it be shown that the algorithm converges to a single answer? Can it be shown that this is the correct answer? In this section these questions are addressed. While it is not possible to prove convergence for MEGA in general, we are able to discuss expectations for the algorithm's performance for certain types of applications.

Global optimization is a difficult problem and also a problem with great breadth. It is often possible to make statements about a algorithm's performance on a narrowly specified subset of all global search problems, but rarely possible to generalize these statements. In fact, it can be shown that algorithms that do not have access to global information can not be guaranteed to converge to a global optimum [54]. Global information may include bounds on function derivatives or other functional forms, and often takes the form of required parameters in an algorithm. The MEGA algorithm specifically avoids requiring this type of knowledge, but clearly does so at the expense of provable definite convergence to a global optimum.

One approach to discussing convergence of algorithms is to examine the likelihood of convergence. Pure random search has been shown to converge to the global optimum with a probability one for every search problem [10, 65]. Additionally, it has been shown that adding mutation operators to controlled random search algorithms can increase the

probability of convergence [35]. However, the provable convergence is weak from a practical standpoint [54, 35], because it does not address the efficiency of the search. The randomness in these algorithms is what makes it possible to prove convergence in probability, but that same randomness can lead to inefficient searches over any finite horizon. Convergence proofs for global optimization not based on random processes are possible but are generally quite limited. It has been shown, for example, that given certain conditions PSO will converge to a single solution [49, 6], but it can not be shown that this solution is the globally best solution.

MEGA belongs to the class of step-size algorithms, along with quasi-Newton methods and sequential random search methods. Under specific conditions sequential random search may be shown to converge to the global optimum with probability one as the iteration counter tends towards infinity [65]. The proof of this assumes that the update procedure chooses the best of the points found so far (is improving), and also that sampling has zero probability of missing any subset $\mathcal{S}_p$ in the search space $\mathcal{S}$. While the former assumption can be ensured in MEGA, the latter is contraindicated by the desire to direct the search step to a promising subset of the search volume. By placing constraints on the stepsize and gradient estimation it can be proved that MEGA will converge on a strictly convex function, but this can not be extended to multi-minima problems.

With the limitations of provable definite convergence, it is interesting to consider the types of search problems for which MEGA was conceived and how the algorithm design interacts with function characteristics. The primary assumption of MEGA is that engineering applications have some structure to them, where structure includes regional trends and limitations on discontinuities. MEGA is designed to use knowledge of that structure to efficiently direct the search. This assumption results in an algorithm design that is expected to perform well on highly structured problems.

In theory the MEGA algorithm can handle all search problems with numerical domain and range. MEGA places no requirements on linearity or continuity, and is specifically designed to work with multi-minima problems. For example, if you consider problems with integer function values a regional gradient will be estimated to connect the integer levels and will point in the direction of the lower solution. In some cases this may fail (such as

if all operating points result in identical output values), but sufficiently distributed initial operating points will prevent this case. Similarly, MEGA will be able to operate on non-differentiable functions by estimating a regional gradient to fit across any discontinuities. A problem with integer inputs is more complicated. While a valid regional gradient will be estimated, the updated operating position position must be mapped to valid input values. In many cases this may be successfully performed with a simple rounding function. The design of MEGA is complete without reliance on specific constraint handling methods. The algorithm works naturally with simple domain constraints, as discussed in the initial design. It can, however, be extended to use other type of constraint handling techniques as discussed in Chapter 4.

There are some applications for which MEGA is not intended. The gradient estimation requires numerical values making this approach unsuitable for problems with categorical inputs and outputs. Discrete domains, as mentioned above, pose some particular modeling challenges. Intuition suggests that problems for which functional information is readily available are better solved using that information directly, while problems that have a nearly random response surface will benefit more from a controlled random search technique.

In some cases the MEGA algorithm may fail to find the best solution to a problem that appears to have qualities that make it well suited to be solved by MEGA. In some cases this happens because the problem has hidden difficulties, such as a very narrow valley containing the global minima. It is easy to picture a function where the global regional gradient points directly away from the global minima, and MEGA will perform poorly in that situation. Additionally, there are failure modes for MEGA that are less problem dependent. Notably, if all the active points converge to a single location (or to a very small region), and the step size is small, it becomes impossible to generate a new point in a different location. If the points converge to a local minimum the regional gradient will not point away from that local minimum, and no points will be generated towards a potential global minimum. In these latter two examples the solution is to maintain a set of points that spans the search space, or to include a method for generating a point in a new location. Including an occasional random step, or using a restart approach can avoid the early-convergence failure mode.

As discussed above, random search techniques can be shown to converge for all problems

in the limit. While no convergence proof exists for MEGA at the moment, convergence could be guaranteed through the extensive use of restart with random initial points. In practice the work here shows that MEGA is often successful in finding a good solution to a variety of problems, and the desire to guarantee absolute convergence through restart must be balanced against the desire for computational efficiency.

2.4 Algorithm Variations

The results of testing the initial formulation of MEGA (denoted MEGA-1 for the purposes of discussing variants) on a set of standard test functions are encouraging. However, testing also reveals areas in which there is some room for improvement. The primary drawback of this new algorithm, as compared to the PSO algorithm, is in the time it takes to compute each new operating point. This time is primarily spent on the clustering step. Two variants are suggested to attempt to address this problem by replacing the clustering step with a different operator. The random resolution (MEGA-RR in section 2.4.1) approach defines the clusters with a proximity function instead of an agglomerative clustering operation. The alternative subdivision method (MEGA-SD in section 2.4.2) uses a more structured approach to evenly divide the active points into sub-sets.

Other issues were also explored. An intriguing question that is yet to be resolved is the impact of the database size and update criteria on algorithm performance. Additionally modifications to the stepsize and boundary handling are shown to significantly affect the performance of the algorithm. Experimentation and improvements in these areas are discussed below.

2.4.1 Random Resolution Method

One primary goal with the MEGA algorithm is to search at multiple resolutions. Motivation and background for this approach is discussed in Chapter 1. A question raised during MEGA-1 development is how to specify the resolutions at which searches occurred. These resolutions should include levels to search the global function terrain, as well as smaller resolutions to search over sub-trends within that terrain. The optimal number of different resolutions required for a given problem depends on the structure of the objective function

[30]. MEGA-1 worked at two resolutions, of which one was defined by the clusters of the active points and the other was defined by the global spread of the active points.

MEGA-RR is created to be more flexible in its resolution definition. In order to allow a formal search on different resolutions we define a regional gradient for the region S_p to be the vector β^* that best fits a hyperplane to the function $f(x)$ over the region S_p. This regional gradient represents the function trend over the region S_p. If one defines the region as a hypersphere centered on $\bar{x}$ and a radius r, one can think of the resolution for this region as being on a scale of r. In the random resolution variant the clustering steps are eliminated, and replaced by a single random-resolution step. Each active point, $\bar{x}$, is assigned a random resolution for the current iteration. The radius r is defined by that resolution which is a random number uniformly distributed between zero and the longest edge of the hyperrectangular search space. A gradient estimation is calculated using all the database points within r of the current active point. Then a search step is taken from the active point $\bar{x}$ in the gradient direction. The stepsize for each search step is defined by the resolution length r as defined in section 2.4.3. This represents the longest step for which the estimated gradient is considered to be valid. The active points are updated in the same manner as in MEGA-1, where a new point is always added and the worst old point is always removed. Additionally, a record of previously evaluated points is maintained. If there are not enough active points for a gradient estimate within r of the current point, points are added from the "old-points" list. Points that are found to be within r of the current point are added until there are $D + 1$ total points.

MEGA-RR represents a more streamlined representation of the fundamental concepts behind MEGA-1. Using random resolutions allows for searches to take place over each reasonable resolution size. Matching the stepsize to the resolution size provides a simple way to base the stepsize on the algorithmic structure. Finally, replacing the expensive clustering step with the random resolution groupings results in a more efficient algorithm. Unfortunately, the MEGA-RR cannot outperform MEGA-1. As seen in Figure 2.11 the convergence rate for MEGA-RR sometimes falls short of the mark set by the original design. Additionally, the mechanism for finding points within r of the operating point negates some of the computation savings of this method.

MEGA-RR allows for a wide array of resolution sizes at every point in the search. It is hoped that this scheme would do better than the possibility of using just one or two, perhaps suboptimal, resolutions. In MEGA-1 the resolution size slowly evolves as the active population of points evolves and draws towards the most promising search areas. This evolution of the population results in reducing the resolution size for more accurate searches at particularly useful locations. It appears that this natural evolution does better at controlling resolution size than a random assignment does.

2.4.2 Subdivision Method

A different variant termed MEGA-SD was designed to decrease the computation requirements of the individual search steps, in order to make a MEGA based search more competitive with other search algorithms. While MEGA-1 outperforms other algorithms in most respects on test functions, it requires more time than other algorithms to find a solution. The most time consuming parts of each MEGA-1 iteration are the clustering operation and the gradient estimation. The gradient estimation is central to the design of MEGA-1, but the clustering operation is only performed as one manner of choosing subgroups of active points. MEGA-SD is created to replace the clustering step with a less expensive operation. In this modified version of the algorithm the active points are subdivided along a line at the median point in each dimension. This operation is extremely fast, and subdivides the active points into smaller and smaller groupings as subsequent dimensions are used.

The algorithm is straightforward and recursively calls a function `subdivide` with all available database points and a specified dimension. `Subdivide` first divides the set of points into half along the median in the specified dimension, and then recalls itself with the two subsets of points and a new random dimension. Each call to subdivide is completed by estimating the gradient for the set of points input to the function, and taking a step in the gradient dimension with a stepsize based on the regional radius for that set of points. Each new point is added to the database, replacing the worst performing old point.

The primary advantage of MEGA-SD is in the computational cost to group the active points into subsets. This does cause other differences, primarily in the way the population

evolves. An advantage of the decreased computational cost, aside from the improvement in the time required, is that more points can be used to estimate the gradient. Additionally, a new resolution is associated with each new subdivision, so more than two resolutions are evaluated. These attributes may be useful, as the overall convergence rate of MEGA-SD is often as good or a bit better than those of MEGA-1. There is significant improvement in the time required for each run, so this variant will prove to be useful for many types of problems.

A comparison between the three MEGA variants may be seen in Figure 2.11. It should be noted that all three of these algorithms used a psuedo-inverse gradient estimation (see Chapter 3). The plots show four different performance measures - the percentage of the test runs that converged (to a previously known global minimum), the average number of function calls required for convergence, the average time required for convergence, and the distribution of minimum values found. The performance of MEGA-1 is as expected, showing moderate convergence measures, but with a lot of time required for the solution. One can see that MEGA-SD is competitive in terms of convergence rate, and is also taking significantly less time to complete a search. MEGA-RR successfully solves some problems that MEGA-1 has difficulty with (such as the Griewank test problem), but does not show a clear gain on any of the performance measures.

2.4.3 Stepsizes

One difficulty revealed by the initial algorithm development is to define a reasonable stepsize. The step length should depend on how far away the search is from a global optimum. Larger stepsizes allow for faster convergence, so long as they are not so large as to overstep the local minimum. An optimal Armijo stepsize is often assumed for convergence proofs. The Armijo stepsize σ is defined for the k^{th} iteration by looking at the derivative of the descent equation. This limits the stepsize based on how quickly the objective function is decreasing. Define ϕ as

$$\phi(\sigma) = f(x_k + \sigma d_k),$$

where d_k is the descent direction. A stepsize σ is considered to be not too large if

$$\phi(\sigma) \leq \phi(0) + \varepsilon\phi'(0)\sigma,$$

where ε is a constant between zero and one. A lower bound on σ is

$$\phi(\eta\sigma) \leq \phi(0) + \varepsilon\phi'(0)\eta\sigma,$$

for a small value $\eta > 1$. This says that if σ is increased by a factor of η it will be too large. In practice the Armijo stepsize is often implemented by assuming an arbitrary value of σ and then repeatedly increasing it by a factor of η until it is too large [31].

The Armijo stepsize may be useful for convergence analysis, but is often not practical for global searches. Aside from the implementation difficulty, global optimization often benefits from larger stepsizes which help to escape from local minima [62]. Progressive or annealing stepsizes are often used in a heuristic method because they allow the stepsize to decrease as the optimum is approached, while still allowing very large stepsizes for a global search. Probabilistic stepsizes also work well with stochastic algorithms [50]. Many of these approaches were tried with the MEGA algorithm during early development work.

A stepsize consistent with the resolutional search philosophy is based on the individual resolution sizes. Many attempts were made to further define and refine this stepsize, and a promising method is implemented. Specifically, the resolution is defined by the point in subspace, $x_{i \in S_p}$. For the region with $\bar{x} = mean(x_{i \in S_p})$ and $\max(\|x_{i \in S_p} - \bar{x}\|)$ as its outer radius, the stepsize is set equal to the outer radius, and the step direction is the gradient estimated over this region. A similar alternative was tried in which $\bar{x}$ was instead defined as the best performer in the subspace.

During the development of MEGA different methods of adapting the stepsize were used, including the geometric progression in the original algorithm description. One of the positive results of using a stepsize based on the resolution of the current gradient is that it is self-adapting. The stepsize always correlates to the size of the region for which the current gradient estimate is valid. Interestingly, this stepsize is self-adaptive in another way because the size of the regions changes as the algorithms progress. In the original MEGA (MEGA-1, as described in section 2.1) the active particles converge towards the promising locations in

the search space. As they converge the regions because smaller causing smaller stepsizes to perform a more accurate search. In the two proposed variants the region sizes also change as the search progresses, but less predictably than in MEGA-1. This interplay shows that it is necessary to pay close attention to how the population evolves. One of the difficulties that arose in the proposed MEGA variants was how to manage the database to result in a useful population evolution.

2.4.4 Database Size and Contents

In the MEGA algorithm there is always a set of 'active' operating points. These active points are those from which regional gradients are estimated, and from which search steps are taken. There is also a database of previously evaluated points which is drawn from when additional information is needed for gradient information (that is, when the gradient estimates would otherwise be singular). The choice of which points remain active, and which points remain in the database can affect how the search proceeds. In MEGA-1 each new point is added to the database, replacing the currently contained point with the worst performance. This maintains a database of a constant size. Adding a new point regardless of its performance allows the search to hill-climb. It also ensures that each iteration has some unique new information which helps avoid early convergence.

It is worth considering that the variants of MEGA that do not rely on clustering do not suggest any specific number of active points. The original active set size $(D+1)^2$ was determined because, since linear regression requires $(D+1)$ points for a unique solution, the clustering algorithm wanted $(D+1)$ points in each cluster and similarly $(D+1)$ clusters for the global estimation. When points are added to the set used for each gradient estimate in a different manner this initial number of active points is not as important. In the random resolution variant, for example, the same point may be used in multiple estimations. Still, it is contended that the number of active points should increase with increasing dimension, and should be at least sufficient to produce more than one unique grouping of $(D+1)$ points. Different schemes for determining the number of active points were examined, but this proved to have only a loose effect on algorithm performance. For this reason, the

variations of the MEGA algorithm still use $(D+1)^2$ points in the initial set of active points.

The total number of active points used in MEGA-1 is practically limited because the clustering algorithm requires more computation for each additional point. This is also true of the proposed random resolution algorithm, where each point is tested to see if it is close to the current operation point. However, the computational costs of the subdivision method do not scale in the same way, and there may be some advantage to having more points with which to calculate gradients. For this reason alternative database management schemes were tried for both variants. As a result of preliminary tests MEGA-RR used a set of active points which were maintained in the same way as those in MEGA-1: new points always replace the worst of the existing points. Additionally, a record of "old-points" was maintained, and used if more points were needed for a full-rank gradient estimate. The subdivision variant stored points in the same manner as MEGA-1. Surprisingly, preliminary tests showed that keeping all the evaluated points available reduced convergence rates. It is hypothesized that this is because new points were often generated in small steps from each other which caused an over-fitting of the regional gradient near the densely located points.

There is another aspect to database management in that the new points generated are the product of the subsets of points used to estimate a gradient and take a search step. MEGA-1, MEGA-RR, and MEGA-SD all use different types of subsets. One consequence of this is that the populations do not converge towards promising search areas in the same way. This appears to have a strong effect on the success of the search, but it is difficult to evaluate because it is implicit in the algorithm behavior instead of explicitly controlled.

Each of the MEGA variants starts out with $(D+1)^2$ operating points randomly drawn from a uniform distribution across the search space. However, the mechanisms for choosing subsets to use in gradient estimations and search steps are different, so the distributions of the populations at the end of an optimization run have different characteristics. Figure 2.12 shows an example of the final configuration of populations for an eight-dimensional quadratic function minimization. These plots show that the MEGA-1 population has converged to a few small locations with many colocated points. This is common behavior for MEGA-1, and may be partially responsible for its success. As the population converges smaller clusters are found resulting in smaller resolution searches. Since stepsize is linked to resolution size,

the stepsize gets smaller near local minima. It is notable that the populations from the random-resolution and subdivision variants do not converge in the same way, and this may be part of the reason that these two approaches converge less reliably.

The converging population is also responsible for some of MEGA-1's frailty because, if all the points move to the same location early, the algorithm becomes trapped in that local minimum. This suggests that there needs to be a balance between population convergence and diversification. Improvements to all three variants may be found by looking at the population management.

2.4.5 Constraint Management

Constrained optimization offers challenges on top of the unconstrained problem. In this work constraints are considered at a very basic level by including a mechanism for handling box-type boundaries. Initially, the constraint violations were penalized using slack variables, a method that forces searches towards the desired feasible regions by penalizing infeasible regions. There are difficulties with penalty methods. A balance must be struck between over and under-penalized states. If there is an insufficient penalty, optimal solutions may be found in infeasible regions. If there is too much of a penalty, the search direction may be more affected by avoidance of the constraint than by the objective function structure. Since the balance was difficult to find in early MEGA testing, later implementations of the algorithm use a different method for boundary enforcement which is described in detail below.

Swarm based algorithms (like PSO) commonly use a repair facility to force active agents from outside the boundary to a point either on the boundary, or inside the feasible region [7]. The initial repair mechanism tried for MEGA mapped an infeasible operating point to the point on the boundary closest to it. Analysis of failure modes in this solution reveal that active points accumulate on a sub-space (as on a boundary line) and prevent the algorithm from successfully searching areas away from that sub-space. Moving points to the interior of the feasible space proved to be more successful. This is done using a reflection method which mirrors points outside the space into the interior. For one dimension at a time:

$$x_d = xmin_d + (xmin_d - x_d) \text{ for } x_d < xmin_d$$
$$x_d = xmax_d - (x_d - xmax_d) \text{ for } x_d > xmax_d$$

This method is similar to the periodic model that was also proposed to eliminate clustering in subspaces during swarm based searches [67] . A more complete discussion of constraint handling options may be found in Chapter 4.

2.5 Algorithm Development Discussion

In this chapter the MEGA algorithm was defined and analyzed. Using the search strategies of estimated gradients, memory, and multiresolution searching, MEGA performs competitively when compared to other state of the art algorithms. The strength of the new algorithm is in its high convergence rates, particularly as the dimensionality of a problem climbs. The weakness of this algorithm is the time that it takes to compute each new search step. A number of modifications to MEGA are discussed, and two new variants are introduced. These modifications are used to demonstrate the interplay of different algorithm components, and their exploration may lead to future improvements on existing algorithms. The variants proposed each address the weaknesses in the original MEGA design, and MEGA-SD proves to be a useful modification. Future exploration of these variants and how they use different search strategies may result in a yet more effective algorithm.

2.6 Using the MEGA Algorithm

This chapter presents the newly developed MEGA algorithm. Users who wish to apply this algorithm should find it a relatively easy task, particularly if they refer to the discussions in this chapter. In section 2.1.5 the reasoning behind algorithm parameter settings is discussed. These parameter settings are based on problem specific values. Further, in section 2.4 variations on the basic algorithm are discussed. This discussion should give users a sense of the algorithm sensitivity to changes in implementation details. Users who refer to these sections should be able to choose parameter settings that make sense for their own problems.

If problems with MEGA performance are uncovered the user is referred to section 2.3 where failure modes are discussed.

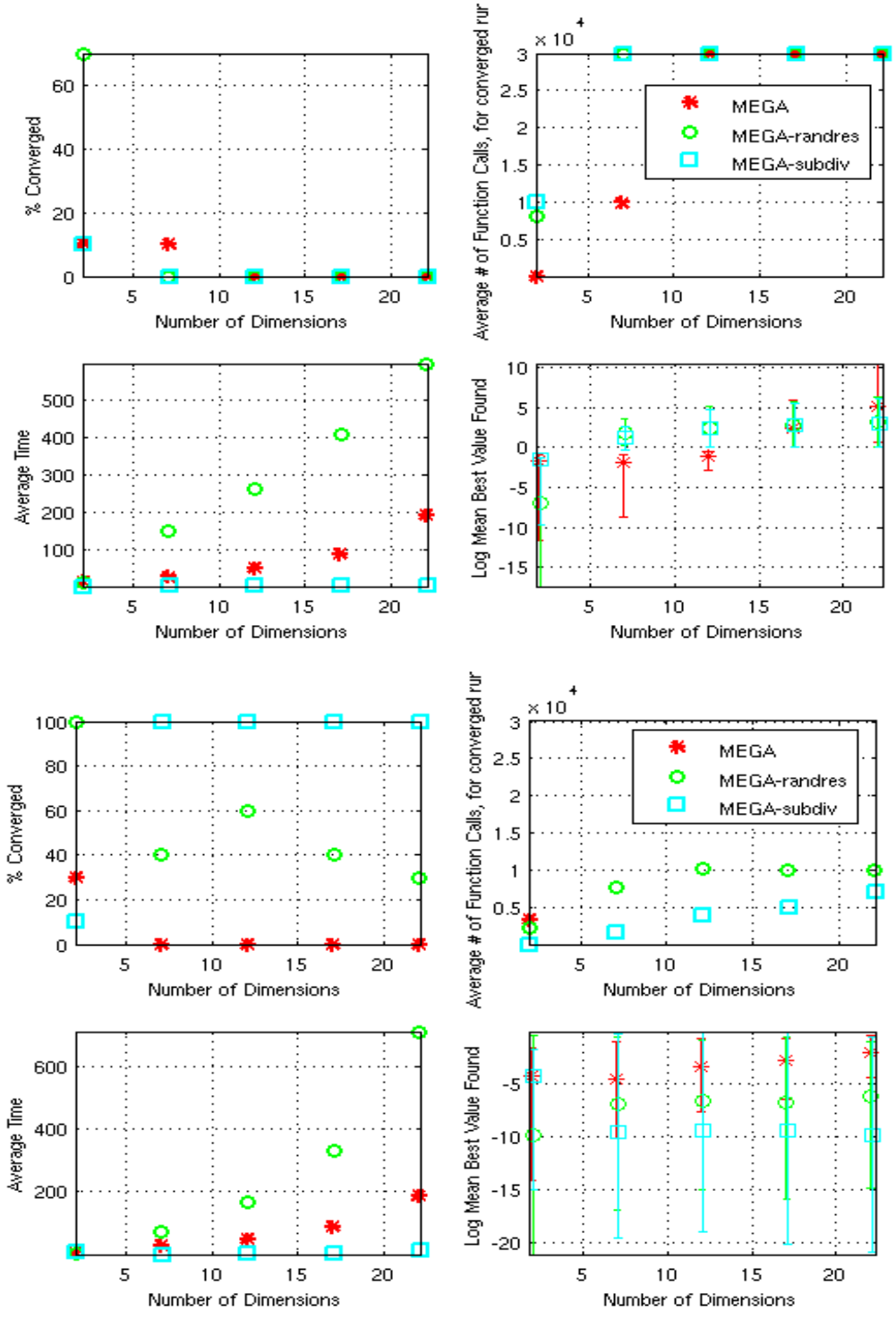

Figure 2.11: A comparison between the three MEGA variants for two different test functions. The top four plots are for the convex Rosenbrock problem, and the bottom four plots are for the non-convex Griewank problem.

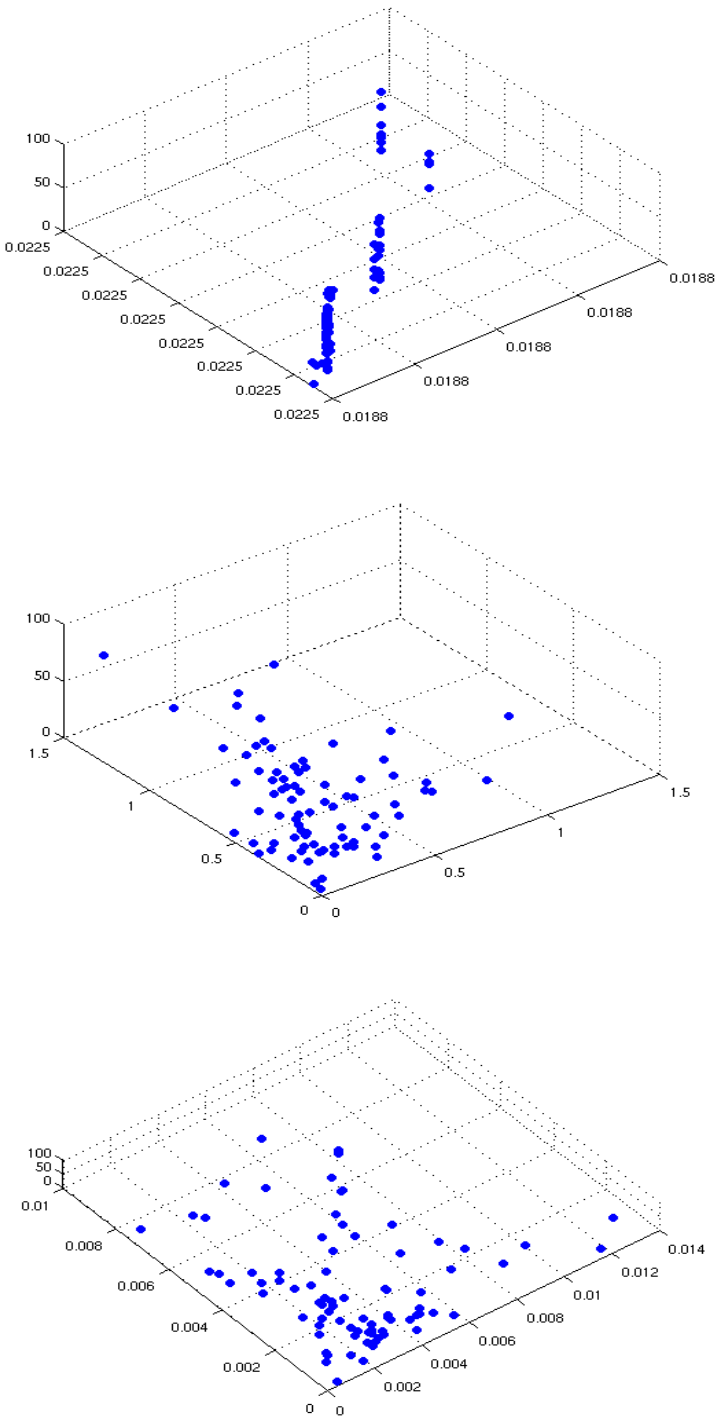

Figure 2.12: Population configuration at the end of an 8D quadratic minimization for three variants of MEGA. Points show f(x) plotted against the first two dimensions of the variable vector. From the top, MEGA-1, MEGA-RR, and MEGA-SD variants. Points are offset vertically to allow visualization of co-located points.

Chapter 3

GRADIENT ESTIMATION AND SEARCH DIRECTION

In this chapter we explore the importance of gradient estimation in directing searches for global optima, as well as how different types of gradient estimates may affect the performance of global algorithms.

A standard algorithmic approach for finding local minima is to calculate or estimate a gradient and follow the path of steepest descent, possibly adjusting for local curvature, as done in Newton-Raphson search. Line search algorithms have been developed for complicated non-linear problems as well. In these algorithms steps are taken in conjugate directions, or descent directions are estimated in trust regions [46]. In global optimization problems line search in a gradient direction may be combined with a method of escaping from, or searching over, local minima so that the global location is found.

Many gradient-free global optimization methods have been developed [44, 65, 4]. Purely stochastic methods are able to converge to a global minimum, but it is our contention that moving downhill can direct the search more quickly toward the optimal area. In fact, many popular global optimization algorithms implicitly or explicitly estimate which direction is downhill at a given operating point. For example, finite difference methods require $2D$ extra function evaluations to estimate a gradient. Simultaneous perturbation stochastic approximation (SPSA) uses a modified finite difference method that approximately estimates the downhill direction using only two extra function evaluations [52]. Trust region optimization uses a linear model of previously evaluated operating points to determine a search direction, and has proven convergence on single-minima functions [46].

In this chapter we focus on two population-based optimization algorithms which incorporate rough estimates of the downhill direction. Particle swarm optimization (PSO) [14], and its variant fully informed particle swarm (FIPS) [33], use a point difference approach to direct the search. The multiresolution estimated gradient architecture (MEGA) [20] al-

gorithm uses a regression-based gradient estimate. FIPS and MEGA are described in depth in Chapters 1 and 2.

In this chapter we first introduce the concept of a regional gradient in Section 3.1 to enable discussion of gradients from a multiresolutional perspective. An experiment compares different gradient estimation techniques for global search using randomly generated sample points in Section 3.2. The more practical question of the impact of different gradient estimates in conjunction with the architecture of a global optimization algorithm is considered for the FIPS and MEGA algorithms in Section 3.3. Section 3.4 ends this chapter with a discussion of the findings and some open questions.

3.1 Regional Gradients and Multiresolutional Searching

A global optimization search happens at many different resolutions. A coarse resolution search finds the most promising area of a large region, while a finer resolution search finds the local minima in a small region. In this section we discuss the challenges of searching at multiple resolutions and define a regional gradient that enables analysis of multiresolutional search.

Some algorithms change the search resolution sequentially: first finding a region of attraction, then searching it locally to find that local optima, then broadening the search again to find another region of attraction. Tabu search describes this process explicitly with the processes of 'intensification' (combining known good solutions to explore their local region), and 'diversification' (searching previously unexplored areas) [15]. Other algorithms accomplish multiresolutional search with a hybrid approach, combining stochastic movement for the global search and more structured gradient descent movements for honing in on optima [63, 45]. Still more global optimization algorithms, such as PSO, search at different resolutions simultaneously, moving the particles toward both locally-promising and globally-promising regions. The MEGA algorithm developed earlier in this thesis alternates between fine and coarse resolution searches by estimating gradients over different regions of the search space.

3.1.1 Regional Gradient Definition

The gradient defines the downhill direction at a particular point in the search space. When the functional space has local minima or noise, the gradient may be ineffective or even misleading as a search direction. Rather, we propose estimating what constitutes the descent direction over different regions of the search space. To this end, a regional gradient is defined for the region S_p to be the vector β^* that best fits a hyperplane to the function $f(x)$ over the region S_p, that is:

$$[\beta^*, \beta_0^*] = \underset{\beta,\beta_0}{\operatorname{argmin}} \int_{x \in S_p} (f(x) - \beta^T x - \beta_0)^2 dx. \tag{3.1}$$

The regional gradient is well-defined whether or not the function f is differentiable. If f is differentiable, then the following proposition holds:

Proposition: If the gradient of $f(x)$ exists and is bounded for all $x \in S_p$, then the regional gradient β^* given in (3.1) is the average of the gradients in the region S_p:

$$\beta^* = \frac{1}{\text{volume}(S_p)} \int_{x \in S_p} \nabla f(x) dx. \tag{3.2}$$

Proof: The proposition follows from the fact that the mean minimizes the integral of squared error if the integral is well-defined. That is, (3.1) is solved by

$$x^T \beta^* + \beta_0^* = \frac{\int_{x \in S_p} f(x) dx}{\int_{x \in S_p} dx}. \tag{3.3}$$

With the propositions assumptions, one can take the gradient of both sides and swap the gradient and integration operators to yield (3.2).

The proposition establishes that the regional gradient β^* is a robust description of the descent direction in the sense that it is the average gradient in the region. In Figure 3.1 two regional gradient estimations are shown for different resolutions. The resolution is determined by size of the marked circular region. The resolution in this description is similar to the trust region used in linear-interpolation trust region methods, although the precise definition of trust regions varies with algorithm specifics [46].

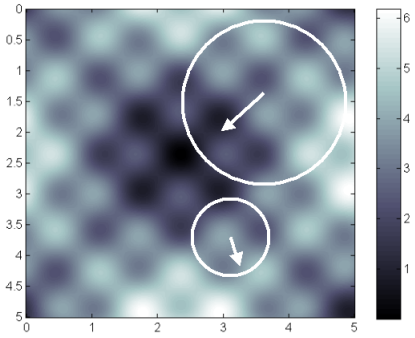

Figure 3.1: This figure shows the sinusoidal test function and two estimated regional gradients of the function (gradient directions are shown as arrows, with regions outlined with circles). The finer regional gradient points towards a local minima, while the coarser regional gradient may help avoid it.

3.1.2 Approximate Regional Gradients in Global Optimization

Many global optimization algorithms can be interpreted as using approximate regional gradients to direct their search, where the regional gradients are calculated for different resolutions.

Simultaneous perturbation stochastic approximation (SPSA) estimates the gradient by stochastically choosing a direction in which to perturb the operating point [53]. Let $\nabla f(x)[d]$ denote the d^{th} vector component of the gradient $\nabla f(x)$. Then the SPSA gradient estimate $\widehat{\nabla f(x_k)}$ for the k^{th} iteration of the algorithm has dth vector component

$$\widehat{\nabla f(x)}[d] = \frac{f(x_k + \delta_k \Delta_k[d]) - f(x_k - \delta_k \Delta_k[d])}{2\delta_k \Delta_k[d]}, \tag{3.4}$$

where δ_k is a scalar that generally shrinks as the number of iterations k increases, and Δ_k is a random D-component perturbation vector. The randomness of the Δ_k vector components adds a stochastic element to the gradient estimations, but does not add bias. It has been suggested that this stochastic element results in a more robust algorithm [32]. The SPSA gradient estimation requires only two new function evaluations, regardless of the dimensionality of the problem. The SPSA gradient estimate can be interpreted as an

estimate of the gradient over a region defined by a hypersphere with center x_k, and radius $\delta_k \Delta_k$.

Point-differences in global optimization can also be interpreted as approximations of the regional gradient. A point-difference is similar to the SPSA estimation, but does not require new points to be evaluated. For example, given two previously-evaluated sample pairs $(x_1, f(x_1))$ and $(x_2, f(x_2))$ where $f(x_2) < f(x_1)$, the point-difference is defined to be the vector $x_1 - x_2$, or the normalized version $(x_1 - x_2)/\|x_1 - x_2\|$. The point-difference estimate is a simple and elegant method of obtaining a notion of "downhill," and is used in PSO. This point-difference estimate can be considered a regional gradient for the local region of x_1, x_2.

Particle Swarm Optimization

PSO is an evolutionary algorithm in which many agents, or particles, search the space collaboratively [14]. The particles' behavior is modeled after the natural flocking behavior of birds. If on the i^{th} iteration the location of a particle is x_i, then on the $(i+1)^{th}$ iteration the particle's position evolves according to the equations

$$\begin{aligned} x_{i+1} &= x_i + v_{i+1}, \\ v_{i+1} &= c_0 v_i + c_1 r_1 (p^* - x_i) + c_2 r_2 (g^* - x_i), \end{aligned} \tag{3.5}$$

where p^* is the current personal best for that particle, g^* is the current global best for all the particles, r_1, r_2 are drawn independently from a uniform distribution over $[0, 1]$, and c_0, c_1, c_2 are scalar parameters. Setting the values of the scalar parameters can be challenging, and there are a number of recommendations for their values. The number of particles used is also a parameter. There is a summary of current PSO information at [1].

In the PSO update equation (3.5), the velocity v_{i+1} specifies the search direction and stepsize for the particle. The velocity is a combination of the difference between the current operating point and the local best, and the difference between the current operating point and the global best. The velocity can be interpreted as a weighted combination of a point-difference estimate of a coarse regional gradient $(g^* - x_i)$ and a point-difference estimate of a generally finer regional gradient $(p^* - x_i)$. One suggestion for improving the canonical

PSO implementation is to replace the point difference local gradient with a finite difference local gradient [41].

In this work, we investigate gradient estimation for a recent variant of PSO called the fully-informed particle swarm (FIPS) [33]. FIPS is structured in a way that makes it more amenable to replacing the point-difference gradient estimates with other gradient estimates, and has been shown to outperform the canonical PSO [33].

The FIPS algorithm defines a fixed neighborhood $\mathcal{N}$ of N nearby particles for each particle. Then the location x_i of a particle at the i^{th} iteration evolves according to the FIPS equations:

$$\begin{aligned} x_{i+1} &= x_i + v_i, \\ v_i &= \chi(v_{i-1} + \varphi(P_i - x_i)), \qquad (3.6) \\ P_i &= \frac{\sum_{k\in\mathcal{N}} \varphi_k p_k^*}{\sum_{k\in\mathcal{N}} \varphi_k}, \qquad (3.7) \end{aligned}$$

where p_k^* is the personal best location of the kth particle at the ith iteration, $\chi = .7298$ and $\Phi = 4.1$. The φ_k are drawn independently and identically from the uniform distribution $U[0, \frac{\Phi}{N}]$, and φ is drawn from $U[0, \Phi]$. These parameter values, suggested variants and a discussion of different neighborhood definitions are given in [33]. In this chapter, the connected ring neighborhood is used, which gives each operating point two neighbors and has been shown to work well [33].

The equations for the FIPS update can be interpreted as an update with a point-difference regional gradient estimate

$$\delta_i = P_i - x_i, \qquad (3.8)$$

where P_i (defined in (3.7)) is a randomly-weighted average of the personal best values in the particle x's neighborhood.

Multiresolution Estimated Gradient Architecture

The MEGA algorithm (described in Chapter 2) was motivated by the idea of gradient descent at multiple resolutions. In MEGA a population of active points evolves. An active

point is used to calculate the next step of the search. The population is initialized at the start of the optimization with $(D+1)^2$ randomly drawn points x_i for $i = 1$ to $(D+1)^2$, which are evaluated to form the initial active population $\{x_i, f(x_i)\}$. Each time a new operating point $(x_{new}, f(x_{new}))$ is evaluated, the current worst point is removed, and then the newly-evaluated point is added to the active population. This replacement forces the population to evolve.

At each iteration a set of line-search steps creates new points. First, fine resolution steps are taken by clustering the active points into $D+1$ clusters of spatially-adjacent points, and taking a step from the cluster center in the direction of the regional gradient associated with that cluster. Second, the $D+1$ new points are used to fit a coarse-resolution regional gradient, and an additional step is taken from the mean of the new points in the direction of that regional gradient.

The regional gradients described in the last paragraph are obtained by fitting a hyperplane to the $\{x_j, f(x_j)\}$ pairs in a given cluster. The squared error is minimized such that the hyperplane has slope coefficients ρ^* such that

$$[\rho^*, \rho_0^*] = \underset{\rho,\rho_0}{\operatorname{argmin}} \sum_{j \in \mathcal{J}_n} (f(x_j) - (\rho^T x_j + \rho_0))^2, \tag{3.9}$$

where ρ_0^* is an offset that does not form part of the gradient direction.

We interpret ρ^* as an estimate of the unknown regional gradient β^* for the region spanned by the clusters' points.

The closed-form solution for ρ^* is

$$\rho^* = (X^T X)^{-1} X^T y, \tag{3.10}$$

where X is a matrix whose jth column is x_j, and y is a vector whose jth element is $f(x_j)$. In practice, we use the Moore-Penrose pseudoinverse to calculate ρ^* such that any singular values of X below a small fixed tolerance value are disregarded. The pseudoinverse provides the unique solution to (3.9) with minimum $\rho^T \rho$. In this way, the pseudoinverse provides a low estimation variance estimate even when fewer than D points are used to fit the hyperplane [19].

3.2 Gradient Estimation Performance

The regional gradient is a way to capture functional trends over a region, and one expects that the regional gradient can be more helpful than the analytic gradient (assuming it exists) when searching for a global optima. To test this hypothesis, we begin with a study of the effectiveness of different gradient estimation techniques abstracted from any specific global optimization algorithm. We compare the angle θ between each normalized gradient or gradient estimate a calculated from a random draw of points to the true direction b to the global optima. That is,

$$\theta = \arccos(a \cdot b).$$

The experiment was run on four different test functions for which the analytical gradient and the global minimum are known. For each run, $(D+1)$ points $x_1, x_2, \ldots, x_{D+1} \in S$ are randomly drawn, and their mean $\bar{x}$ is calculated. The $D+1$ points are drawn uniformly across the hypercube defined by the domain of the test function. However, the maximum difference between the points in any single dimension is limited to a fraction of the entire domain, and any points exceeding that distance from the original set are thrown out. This fraction limits the region for the gradient estimate, and is set to 0.9 for the results in Tables I and II. The analytical gradient is taken to be the true gradient at $f(\bar{x})$. The regional gradient is calculated over a region defined by a hypersphere with $\bar{x}$ as its center and $\max(\|x_i - \bar{x}\|)$ as its radius, and the formula (3.1) is approximated by a least-squares hyperplane fit to 1000 points drawn uniformly over the region.

A pseudoinverse regional gradient estimate ρ^* is calculated as in equation (3.10) for each draw of $D+1$ points. A second regional gradient estimate is formed for each draw of $D+1$ points by the coefficients of a regularized least squares fit of hyperplane coefficients. Regularized least-squares linear regression penalizes the slope of the fitted hyperplane, in order to reduce estimation variance (that is, the estimates tend to be less variable across different draws of the $D+1$ random points). We regularized using ridge regression [19, 21], such that the estimated coefficients ϱ^* are defined by

$$[\varrho^*, \varrho_0^*] = \underset{\varrho, \varrho_0}{\operatorname{argmin}} \left\{ \sum_{i \in \mathcal{J}_n} \left(f(x_i) - \varrho^T x_i) - \varrho_0 \right)^2 + \lambda \varrho^T \varrho \right\}, \tag{3.11}$$

Table 3.1: Angle θ values for single minima test functions

	2D Quadratic	2D Rosenbrock	10D Rosenbrock
analytic gradient	0.000	0.827	0.562
regional gradient	0.035	0.831	0.852
pseudoinverse	0.646	1.096	1.133
ridge	0.541	1.054	0.879
point-difference	0.673	1.198	1.039

where λ is a parameter controlling the regularization. Based on preliminary experiments, a fixed value of $\lambda = .01$ is used throughout this work. A third estimate of the regional gradient is a point-difference (PD) which is designed to emulate the PD estimate found in the FIPS algorithm (see equation (3.8)), such that

$$\begin{aligned} PD &= \bar{x} - x_{i^*}, \\ i^* &= \operatorname*{argmin}_i f(x_i). \end{aligned} \tag{3.12}$$

3.2.1 Comparison of Estimated Gradients via Simulation

Results of the average values of the angle θ to the optimal direction are given in Tables 3.1, 3.2, and 3.3. The statistical significance of these results is analyzed using the Wilcoxian signed rank test, and all cases are found to be statistically significantly different at a significance level of 0.05. Note that the value of θ can range from $[0, \pi]$, and the expected value of θ for a randomly directed vector is $\pi/2$, or 1.57. Thus even for the sinusoidal and Griewank functions with many multiple minima, all of the search directions are better on average than random.

Table 3.1 gives the results for two monotonic functions. Even for monotonic functions, the analytic gradient may be misleading if the curvature varies over the range of the function. For example, in Rosenbrock's function (Figure 2.1) the analytic gradient only correlates loosely with the true direction to the optima. Table 3.1 shows that in these convex function

cases, the regional gradient is a reasonable approximation of the analytical gradient, and the point-difference and two hyperplane fits provide similar approximations.

One expects the regional gradient to be the most useful when the test function has a strong global trend with weak local fluctuations, such as the Griewank function (Figure 2.1). We expect regional information to be less useful when the local trends are just as strong as the global trends, such as in the sinusoidal function (shown in Figure 3.1.) However, even in such cases, we believe that the regional gradient will provide more useful information than the analytic gradient, which is more likely to point one toward a local minima. In fact, this hypothesis is confirmed by the results in Table 3.2, which shows the regional gradient to be the best indicator of the optimal direction for the sinusoidal and Griewank functions. As expected, the regional gradient is most effective for the Griewank.

The gradient estimates perform similarly to each other, but relatively poorly, on the sinusoidal function. On the Griewank function the point-difference is significantly worse than the hyperplane estimates, suggesting that there is too much error in this estimate. For the Griewank function calculated over a ten-dimensional hypercube, the global trend is accentuated and the regional gradients show an additional advantage. Throughout, the ridge regression estimate is shown to be slightly more effective than the pseudoinverse or point-difference in two-dimensions. It is much more effective for the ten-dimensional case. Estimation error is either due to estimation variance (which describes how variable the estimate is when the sample points change) or estimation bias (which describes how wrong the average estimate is when the average is over many random draws of sample points) [19]. Given that there are only $D + 1$ points in D dimensions used for the gradient approximations, estimation variance will generally be a greater problem than estimation bias. The ridge regression's estimation variance reduction is what causes it to be the best estimator. Similarly, the pseudoinverse consistently performs better than the point-difference. This difference is because the point-difference has the highest estimation-variance due to its use of the minimum $f(x_i)$, which can vary greatly for different draws of $D + 1$ points.

Next, we present data on how the size of the region used to estimate the gradient impacts the metric θ. Table 3.3 compares θ values for the 2D Griewank function where the resolution of the gradient was limited by setting the maximum distance between any two points in the

Table 3.2: Angle θ values for multi-minima test functions

	2D sinusoidal	2D Griewank	10D Griewank
analytic gradient	0.294	1.339	0.094
regional gradient	0.211	0.014	0.034
pseudoinverse	1.078	0.373	0.900
ridge	1.038	0.333	0.573
point-difference	1.180	0.608	0.957

Table 3.3: Angle θ values for different resolutions in the Griewank test function

	Resolution as % of domain		
	0.2	0.5	0.9
analytic gradient	1.298	1.332	1.339
regional gradient	0.008	0.022	0.014
pseudoinverse	0.289	0.483	0.373
ridge	0.277	0.395	0.333
point-difference	0.639	0.642	0.608

test set was limited to 0.2, 0.5, and 0.9 of the function domain for each dimension. Note that the analytic gradient does not depend on the region size, and so the analytic gradient row gives an indication of the variance of the results. The results show that, for the Griewank, changing the resolution has a large impact on the power of the regional gradient. The impact on the three $D+1$ point gradient estimates is smaller, perhaps because their fidelity to the true regional gradient is limited already. In the context of a practical global optimization algorithm, the regional size used to estimate gradients will vary, either due to chance or by design. Since some region sizes may result in more useful search directions, a well-designed region size variation may be quite advantageous.

3.3 Regional Gradient in Evolutionary Algorithms

Gradient estimation is only one part of the overall behavior of a global optimization algorithm. The algorithms considered here are population-based, which means that the methods for choosing and evolving the population of operating points may have a profound impact on the overall performance of the search. The population distribution directly impacts the regions over which gradients are estimated, thereby dictating how the current set of operating points is updated. At the same time, the population must maintain a record of the most promising points, and allow some hill-climbing in the search.

Here, a controlled comparison is made between different gradient estimates in the context of FIPS and MEGA. The results indicate how important the quality of the gradient estimate is to the final performance of the algorithms. In these tests each algorithm is allowed to run for 10,000 function evaluations so that different configurations and algorithms can be compared.

3.3.1 FIPS

The FIPS update equations result in a velocity (v_i), which includes both the direction and the stepsize for a given step. To control the FIPS step-size across the different gradient estimates, the original velocity magnitude is used for all the gradient estimates. The original FIPS point difference estimate can be viewed as a regional gradient for a region centered

on the midpoint between x_i and P from equation (3.7), so the other gradient estimates are calculated for the same region. The analytical gradient is also evaluated at that midpoint.

The original FIPS point-difference is expected to perform well, but I hypothesize an improvement when better gradient estimates are used. The results in Figures 3.2 and 3.3 show the average minimum value found for 100 algorithm runs, on a log scale. The error bars mark the 25th and 75th percentile of the minimum values. The gradient calculation techniques are the analytical gradient, the estimated regional gradient from equation (3.1) (betastar), the ridge regression gradient estimate from equation (3.11) (ridge), the pseudoinverse gradient estimate from equation (3.10) (pinv), the FIPS gradient estimate from equation (3.6) (point difference), and a randomly generated search direction (random). With the exception of the Rosenbrock function and some dimensions of the sinusoidal function, the analytical and regional gradient outperform all other options. Using a random direction for the gradient performs the worst in all cases, and gives an indication of the importance of gradient information to the population-dynamics.

For the Rosenbrock function it is known that the gradient information does not always supply the most direct route to the global minimum (see Table 3.1). I hypothesize that well-designed population evolution and parameter choices may compensate for poor gradient information. Since FIPS is designed to work with a point-difference gradient, the step-size and parameter settings may be suboptimal for the other gradient estimates and actually result in a misdirected search. This may be particularly true if the regional gradient estimates are misleading. This may be the case in some of the sinusoidal test function runs.

The experiments of Section 3.2 showed that the ridge regression could produce better gradient estimates than the pseudoinverse or point difference. However, when coupled with the FIPS algorithm the ridge estimate ϱ lags slightly behind the pseudoinverse estimate ρ. Possibly the estimation bias of the ridge estimate ends up slowing convergence, or the reduced estimation variance actually reduces the randomness of the search and hurts the progress of finding the optima. It is also possible that the fixed ridge regression parameter $\lambda = .1$ is too high or too low for the different cases encountered in the optimization. A better scheme might be to adapt the regularization parameter λ based on the step-size, the size of the region spanned by the sample points, or on the size of the errors in the

least-squares hyperplane fit. In this case the extra parameter becomes a liability and ridge regression becomes less attractive. Optimizing the FIPS parameters for the pseudoinverse or ridge estimate would likely increase performance.

A difference between the FIPS point-difference and the other estimates is the randomness. It is thought that the PSO algorithms work by driving particles toward, but not directly to, the nearby minimum. This process of overshoot ensures that the algorithm does not get stuck in local minima. Since earlier analysis showed that the point difference estimate is not as good as other techniques for pointing directly to the global minima, the randomized misdirection appears to help FIPS avoid local minima. In fact, the FIPS point-difference shows a generally larger improvement over the pseudoinverse and ridge regression estimates for the two functions with multiple minima than it does the two convex functions.

3.3.2 MEGA

The MEGA algorithm is designed to explicitly use gradient estimates. Therefore we expect that the higher fidelity gradient estimates will improve the overall performance. For the MEGA experiments, the point-difference estimate is given in (3.12). The stepsize is the maximum size of the cluster in any dimension ($\max_d(x_{max}[d] - x_{min}[d])$), and the gradient estimation techniques are used only to determine the search direction.

The results from the MEGA algorithm are shown in Figures 3.4 and 3.5. The figures show the average minimum value found for 100 algorithm runs, on a log scale. The error bars mark the 25th and 75th percentile of the minimum values. The gradient calculation techniques are the analytical gradient, the estimated regional gradient from equation (3.1) (betastar), the ridge regression gradient estimate from equation (3.11) (ridge), the pseudoinverse gradient estimate from equation (3.10) (pinv), the point difference gradient estimate from equation (3.12) (point difference), and a randomly generated search direction (random).

For all the functions but Griewank, the analytical gradient outperforms the other gradients. The regional gradient, and then the pseudoinverse gradient, also perform well. In some of the dimensions of the sinusoidal test function the regional gradient estimate performs poorly, suggesting that this estimate offers a misleading search direction. Similar

behavior was seen in the FIPS results. One reason for this result may be that the regional gradient is being estimated at the wrong resolution.

In the Griewank test function – the one most suited to the use of a regional gradient – the regional gradient and the pseudoinverse estimate perform the best. For the MEGA results, the random descent direction and point-difference estimated gradient do relatively poorly. As with FIPS, the ridge regression does not do as well as the pseudoinverse; we have discussed possible reasons for this in Section 3.3.1.

3.4 Gradient Estimation Discussion

The primary question investigated in this chapter is how regional gradient estimates affect the performance of global optimization algorithms. The results presented in Section 3.2 show that a regional gradient can serve as a pointer to the global minimum in a multimodal function. In Section 3.3 it is shown that gradient-based steps do improve the performance of two population-based search algorithms over random steps. The functional trend is strongest in the Griewank function, and for both algorithms the regional gradient proves advantageous over the analytical gradient for this function.

The random search direction experiments indicate the utility of using the gradient approximations. There is generally a large difference between the analytic gradient results and the random results. This shows that the search direction is important. However, there is a surprising amount of overlap there between the minimum values obtained with the random search direction and the gradient-approximation search directions. This may imply that the quality of the gradient estimates is too poor to be useful. It may also be that the population evolution alone can move the search close to the goal, or that randomness in the search direction is useful in its own right for escaping local minima. The error introduced by the gradient estimates tends to be correlated between consecutive iterations, and may not be random enough to offer the advantage of the pure random search directions.

It is well-known that randomness in a global optimization algorithm can be useful, proven at one extreme by the fact that pure random search will converge to the global minima while many non-random search strategies will become trapped in local minima. FIPS can be considered a *controlled randomness* algorithm, while MEGA has no randomness apart from

the initial draw of points. We expect that the greater randomness in FIPS will make it more robust to objective functions with little structure, while MEGA will perform better when there are strong regional trends toward the global minima. For example, on the Griewank function, MEGA (with the pseudoinverse) achieves an average minima value of about 10^{-5} for all dimensions, whereas FIPS's average minima value is above 1 for dimensions 12 and higher.

On the other functions there are similar performance trends, with the FIPS average performance degrading as the number of dimensions rises, but the average MEGA performance less dependent on the number of dimensions. Although each algorithm is allowed 10,000 function evaluations per run, each run of MEGA takes longer than FIPS due to MEGA's clustering step. If the comparison is done giving each algorithm a total computation time then the results might differ. However, for many practical problems, evaluating the objective function requires running a simulation or other time-consuming processes, and so the cost of each function evaluation may be the most time-consuming part of the optimization, overshadowing the computational differences between FIPS and MEGA.

The results in Figures 2 and 3 prompt consideration of the theoretical performance of the two algorithms: given a perfect analytic gradient, how well can these population-evolution architectures perform? The quadratic and Griewank functions show little difference between the two algorithms, but the Rosenbrock function is easier to optimize given analytic gradient information with the MEGA population-dynamics. For the multiple minima sinusoidal function the algorithms perform comparably until the 22 dimension case is reached, at which point FIPS performance becomes much worse than MEGA.

Given random search directions the FIPS algorithm has a better best case solution in two dimensions, but generally does worse in higher dimensions across all four test functions. When we control for the gradient direction, the difference between the two algorithms is the population evolution mechanics, so we may hypothesize that the MEGA evolution is more useful in higher dimensions.

Population dynamics determine which points are available for estimating gradients, and those points determine the regional span of the gradient. More important for MEGA than FIPS is that $D + 1$ linearly independent points are available for each estimate. Neither

MEGA nor FIPS exert much control over the evolution of the regional sizes or explicitly maintain populations that form a linearly independent span of the space. In the trust region methods presented by Powell [46] some of the optimization steps are designed to minimize the function and some are designed to ensure that the simplex used for calculated gradients is not degenerative. Algorithms such as FIPS and MEGA might similarly benefit from occasional steps to ensure that the population is well-distributed in the search space.

One challenge for global optimization algorithms is the definition of the size of each search step. Current methods for determining stepsize vary, often including a random element, or adapting according to whether the previous step returned a promising solution. In both FIPS and MEGA the step-size is a function of the size of the region represented by the regional gradient estimate, such that the step-size makes appropriate use of the regional gradient estimate. Further advances in global optimization may be possible by considering the stepsize and region size to be linked.

Gradient-based optimization algorithms can be shown to converge for functions that only have one minimum [44], but proving convergence for global optimization is difficult. One approach may be to consider functions that have smooth regional gradients at some resolution, and prove convergence at that resolution.

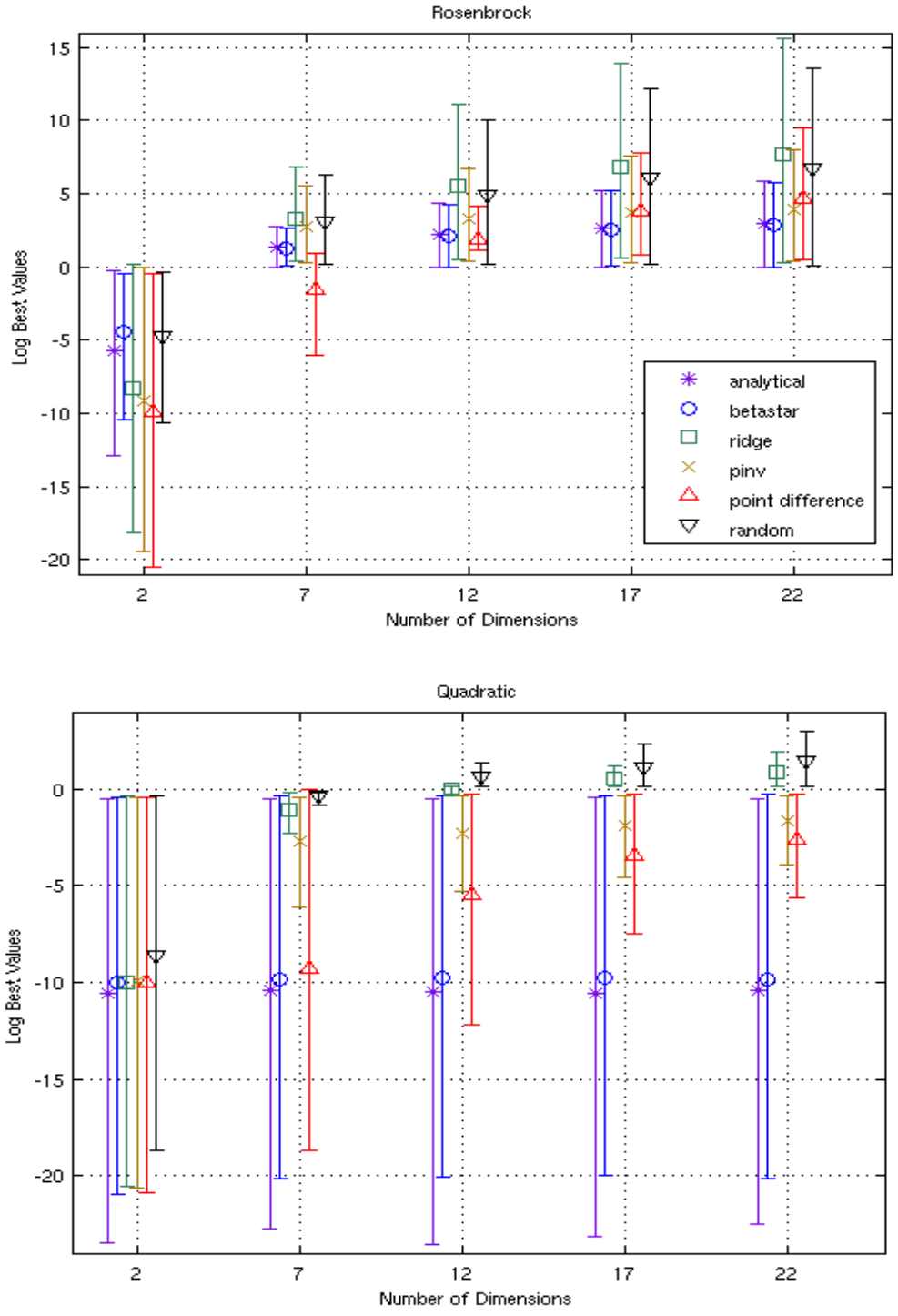

Figure 3.2: The FIPS results for the single minimum test functions (Rosenbrock and quadratic).

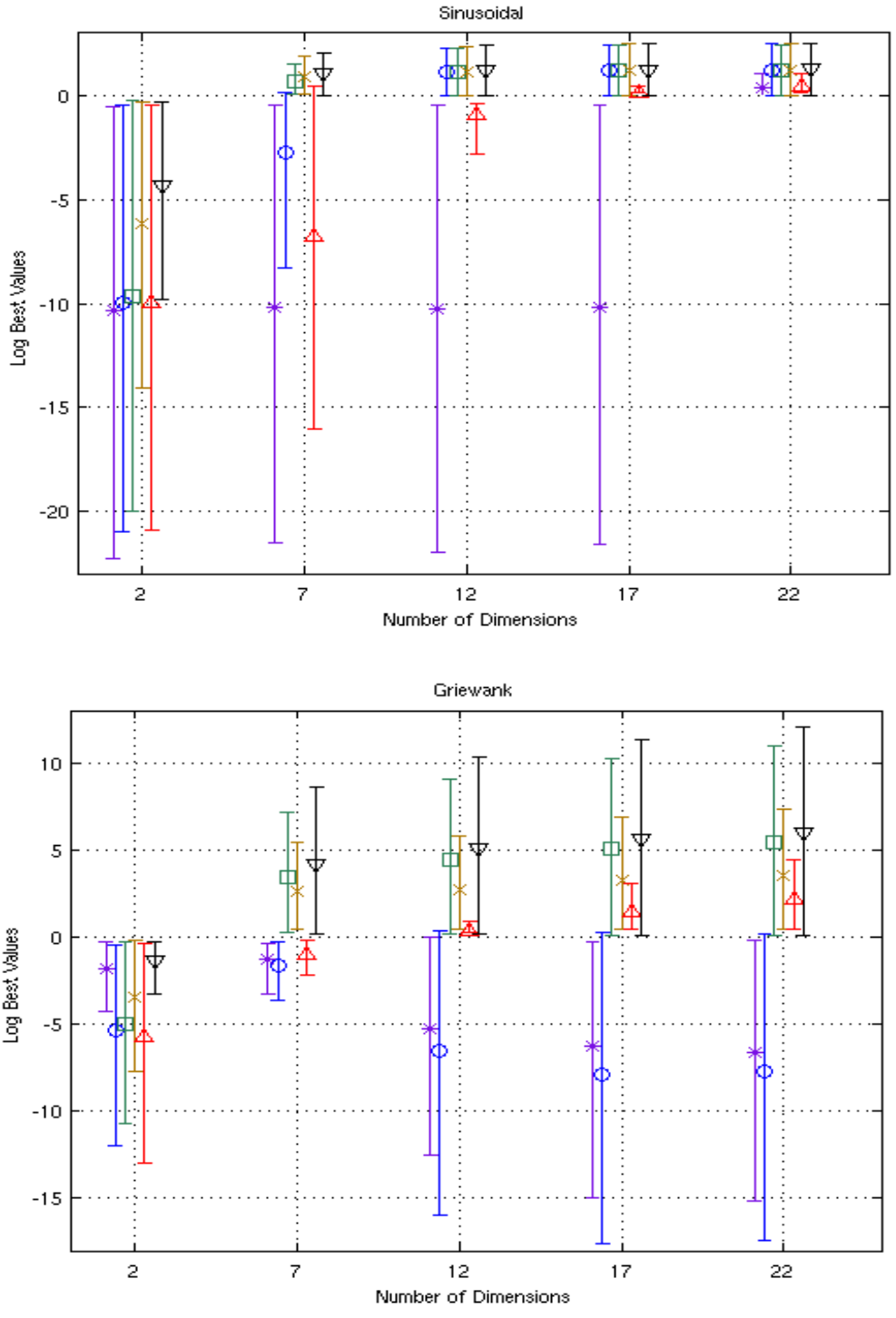

Figure 3.3: The FIPS results for the multiple minima test functions (sinusoidal and Griewank).

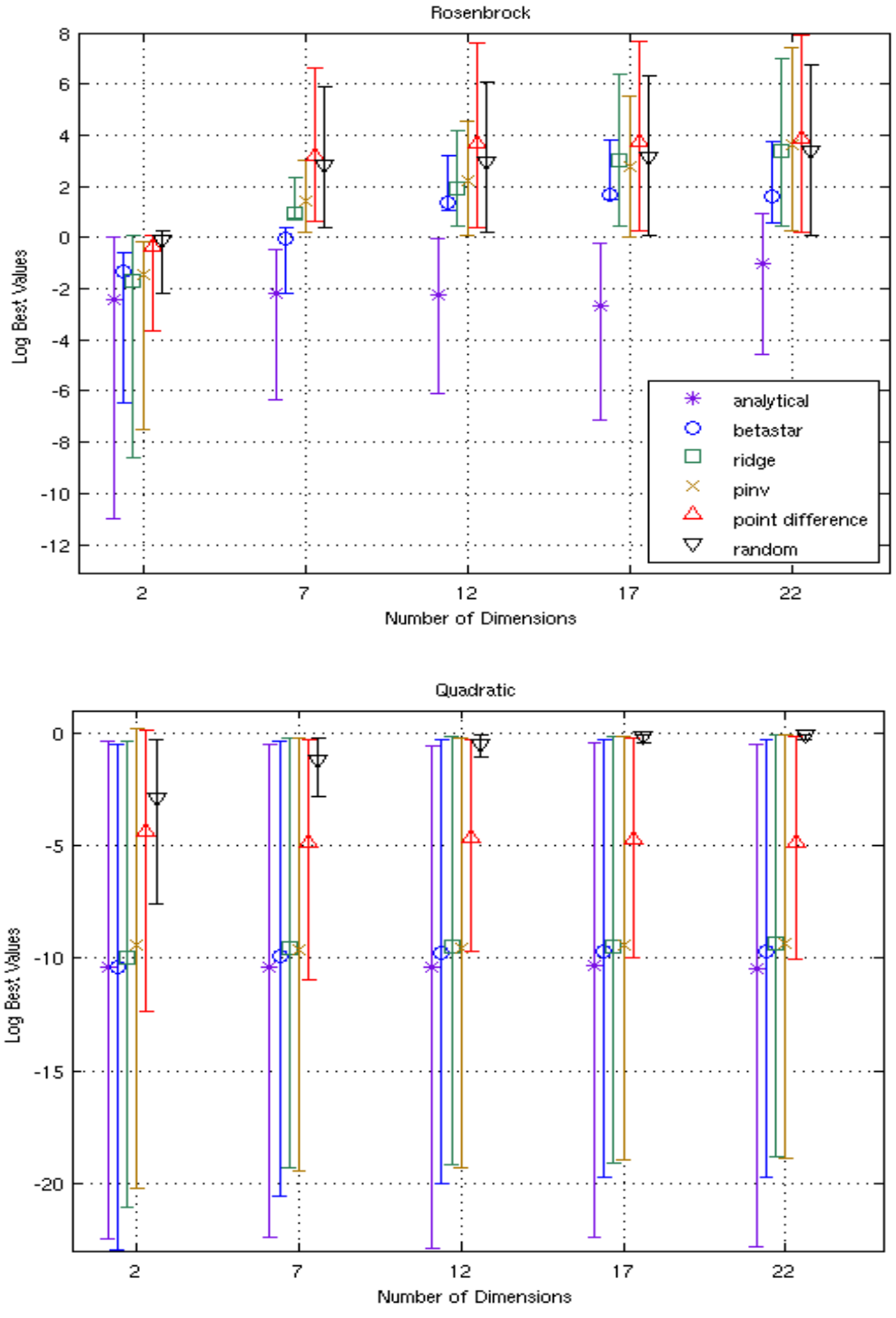

Figure 3.4: The MEGA results for the single minimum test functions (Rosenbrock and quadratic).

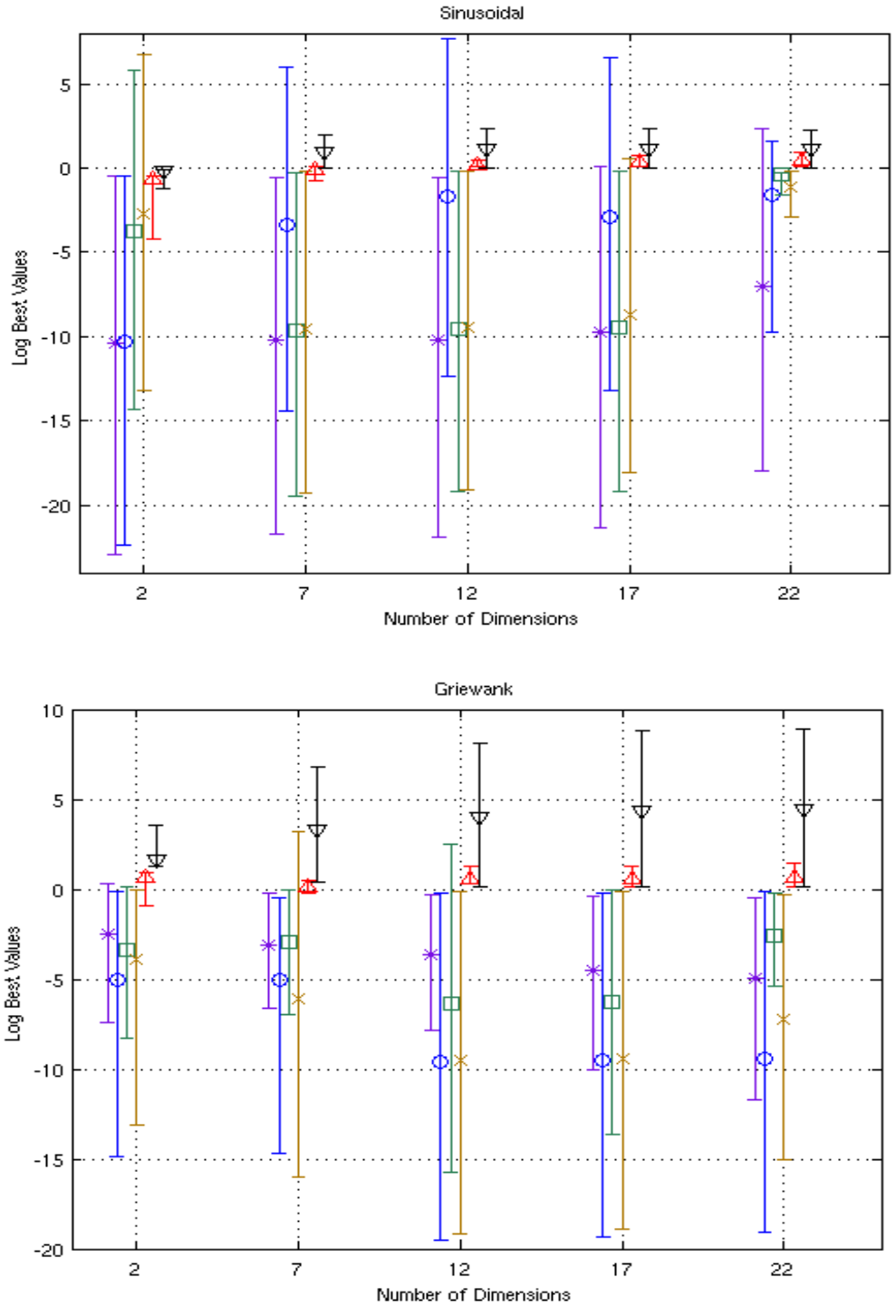

Figure 3.5: The MEGA results for the multiple minima test functions (sinusoidal and Griewank).

Chapter 4

CASE STUDY: GLOBAL OPTIMIZATION FOR SIMILARITY-BASED CLASSIFICATION

The true difficulty, and interest, of global optimization lies in its application to a real problem. Application is usually non-trivial, as each problem has its own characteristics and poses its own particular difficulties. During the course of this research a few different applications were considered, focusing primarily on those where the objective function evaluation was computationally expensive. In this work MEGA was compared to one or more PSO variants. PSO was chosen for comparison because it is a competitive modern search technique and is often used for the types of problems for which MEGA is designed. In the transmitter localization problem discussed in section 4.4 a solution was found using a convex optimizer with multistart (specifically expectation-maximization) worked as well or better than either of the global optimization approaches [25, 39].

This chapter looks at using MEGA to fit weights for a similarity-based nearest-neighbor classification scheme. The objective function for this problem is highly structured and has a dimensionality based on the classifier neighborhood size. Tests show that, for cases where the neighborhood is large, MEGA can provide better classification accuracy. This finding agrees with the test problem simulations which show that MEGA has a greater advantage in high-dimensional searches and when the problem has structure. The problem is described in section 4.1. The weighted nearest neighbor classification structure offers a highly structured nonconvex search problem as well as new challenges in constraint handling. Previous work with MEGA considered only domain limits, and particularly box constraints. This work includes a more complex equality constraint because the weights must sum to one. A discussion of constraint handling variations is found in section 4.2.

4.1 Similarity-based Nearest Neighbor Classification

Cazzanti and Gupta describe similarity-based learning as follows [5]: *Similarity-based learning methods make inferences based only on pairwise similarities or dissimilarities between a test sample and training samples and between pairs of training samples. The term similarity-based learning is used whether the pairwise relationship is a similarity or a dissimilarity. The similarity/dissimilarity function is not constrained to satisfy the properties of a metric. Similarity-based learning can be applied when the test and training samples are not described as points in a metric space. This occurs when the samples are described as feature vectors but the relevant relationship between samples is a similarity or dissimilarity function that does not obey the mathematical rules of a metric. Another case is when the samples are not described as feature vectors at all, but pairwise similarity or dissimilarity information is available. Such similarity-based learning problems arise naturally in bioinformatics, information retrieval, natural language processing, and with geospatial data.*

In this chapter, weighted nearest-neighbors are used for similarity-based learning. In weighted nearest-neighbors, one is given a set of training samples $x_i \in \mathcal{R}^d$ and corresponding class labels $y_i \in \mathcal{G}$ for $i = 1, \ldots, n$. For example, the training samples might be emails, and the class labels might be *spam* and *not spam*. A class $\hat{y}$ is assigned to the test point x based on class labels of the k nearest neighbors:

$$\hat{y} = \arg\max_{g \in \mathcal{G}} \sum_{i=1}^{k} w_i \delta(g, y_i), \tag{4.1}$$

where $\mathcal{G}$ is the set of class labels and δ is the Kronecker delta. The weights w_i are chosen from the set $0 \leq w_i \leq 1$ such that $\sum_{i=1}^{k} w_i = 1$. These limits allow the weights to be viewed as a posterior probability density function for the classes, which makes the classifier flexible: posterior distributions allow class probabilities to be determined, and class probabilities can be used in situations where there are other probabilities to be considered in the problem or asymmetric costs to misclassification. A challenge in developing an effective k-NN classifier is to determine the optimal weights for each set of neighbors. This chapter focuses on using global optimization to choose a set of weights. The objective function for this project was recently proposed by Yihua Chen for similarity-based classification:

$$\begin{aligned} \min_{w \in \mathbb{R}^k} \ & \frac{1}{2} w^T S w - s^T w + \lambda w^T w \\ \text{s.t.} \quad & w \succeq 0, \quad \mathbf{1}^T w = 1. \end{aligned} \tag{4.2}$$

In this function S is the $k \times k$ matrix containing the pairwise similarities for the training samples where $S_{ij} = s(x_i, x_j)$, s is the vector with the ith component $s_i = s(x, x_i)$, and λ is a regularization parameter. This function is a generalization of a regularized linear interpolation objective function proposed by Gupta *et al.* [17]. The three terms of this objective function are used to balance the weight between counting very similar training points and not over-counting repetitive training points. The $s^T w$ term alone would put all the weight on the training point most similar to the test point. It is regulated by the weighted variance term ($\lambda w^T w$). Minimizing the variance rewards spreading the weights evenly across all the training points. The quadratic term $w^T S w$ increases when the similarity between two training points is high, so minimizing it results in weights that place more emphasis on training points that are more dissimilar to their other neighbors.

When the similarity matrix is positive-definite, optimizing the weights is a straightforward quadratic programming problem. This type of problem can be solved in polynomial time with the ellipsoid method. In some situations, such as those with asymmetric similarities, the matrix is indefinite, and the problem is no longer convex. One method of dealing with this problem is to force the matrix to be positive definite by modifying the eigenvalues so they are all positive. This can be done, for example, by shifting up the spectrum or clipping negative eigenvalues. In this work we directly solve the optimization problem using global optimization to solve the original, possibly nonconvex, problem given in (4.2). It is hoped that using the unmodified similarity matrix will result in better classification results. MEGA should perform well in this objective function, which has strong trends but it not necessarily convex (see section 4.3).

4.2 Constraint Handling

Constraint handling is an important, and often overlooked, aspect of applying an optimization algorithm to a real-world problem. Constraints restrict the legal solutions to some

$\mathcal{S} \subset \mathcal{R}^D$, and optimization algorithms must find an acceptable minimum within the resulting feasible set. Constraints may limit the legal range for one or more input parameters, limit the legal operating space to a subspace of the entire search space, or simply define each operating point as legal or illegal with a black box function. Additionally, with open constraints the objective function will return an answer outside the feasible region, but closed constraints exist where there is no solution to the function in the infeasible region.

The method for handling the function constraints is highly dependent on the type of objective function and the type of constraints. Approaches include penalty functions, repair algorithms, special operators (which ensure generation of feasible solutions), and hybrid methods [7]. Another taxonomy is created by considering how the constraints are used. Constraints may be used only to evaluate if a point is feasible (resulting in a death penalty or repair scheme), combined into the objective function (resulting in a penalty scheme), or used in a multi-objective optimization (where the goal is to minimize both the objective function and the constraint violation) [55].

Each of these approaches has relative merits and drawbacks. Multi-objective approaches use the tools developed for multi-objective optimization, and treat minimizing constraint violation as a second objective. These do not guarantee feasibility, and the tradeoff between minimizing the objective function or the constraint function must be addressed. In order to ensure feasible solutions the search must be additionally biased towards the feasible region [48]. Hybrid systems are often two step approaches [55], searching the constraint space and objective spaces separately. One might have different rules for dealing with a feasible operating point than an infeasible one [60]. In other cases proxy functions or relaxation methods are used [61]. Hyrbid systems can be complex to design and implement. Generative solutions are one powerful way to deal with constraints because they result in the entire search existing in the feasible region. In a generative solution the method of generating the next operating point always results in a feasible point. They may be a necessary approach in the case of closed constraints. However, generation functions must be created freshly for each new problem, and they may be difficult to design. An alternative similar to generative functions is to use a death penalty approach, as in [24]. In this approach each new operating point is either feasible, or discarded. Since each infeasible point is discarded, death penalty

approaches also ensure that the entire search happens in the feasible region. In cases where this region is small they can hamper the search.

Penalty functions are commonly used, because they are easy to implement. In these approaches a penalty for violating the constraints is added to the objective function value. This penalty makes the infeasible region unattractive to optimization algorithms. Repair algorithms are also commonly used. Here, an operating point in the infeasible region is mapped into the feasible region.

In this thesis consideration of constraints is focused on inequality and equality constraints:

$$\min_{x \in \mathbb{R}^D} \; f(x)$$

such that

$$g_i(x) \leq 0, i = 1 \ldots n,$$
$$h_j(x) = 0, j = 1 \ldots p.$$

Previous work with MEGA considered only boundary type constraints of the type $l_i \leq x_i \leq u_i, i = 1 \ldots D$. This box constraint, a subset of the inequality constraints, is very common. Both penalty functions and repair functions have been used with MEGA to handle the boundary constraints. In Chapter 2, a system of slack variables is described. This is representative of a penalty approach. Penalty approaches can be difficult to manage because the penalty must be defined by the user. Over-penalizing constraint violation limits the exploration ability of the algorithm, while under-penalizing the violation may be insufficient to lead the search to feasible regions. Novel approaches to manage this conflict include annealing-type changes to the penalty, adaptive penalties, and even a stochastic type of penalty [34]. A repair approach was adapted for MEGA, and is described in the section on variants in Chapter 2. The primary difficulty with repair techniques is in developing the heuristic to map an infeasible point to a feasible point.

The similarity-based classification problem offers a new challenge in constraint handling, because the sum of the w vector is limited by an equality constraint: $\sum_{i=1}^{k} w_i = 1$. Previous

implementations of MEGA dealt only with a boundary constraint, and used either a slack variable based penalty method or a repair function to ensure that the search stayed within the boundaries. Details of this constraint handling may be found in Chapter 2. Both of these methods were used for the classification problem, to ensure that the search remained on the simplex defined by the equality and boundary constraints. The choice of a penalty function for solutions where $\sum_{i=1}^{k} w_i \neq 1$ is quite difficult. The feasible region is quite narrow compared to the search space so a large penalty is required to remain on the desired simplex, but a large penalty also overwhelms the gradient needed to minimize the function. A repair scheme also caused some difficulty, because normalizing the weight vector to sum to one resulted in new operating points that were not in the gradient direction from the old operating points which slowed the search. Another approach considered for the equality constraint was to solve the problem for $k-1$ weights and set $w_k = 1 - \sum_{1}^{k-1} w_i$, as proposed by AlRashidi and El-Hawary [3], but this proved difficult to integrate with the boundary constraints.

The solution used in the results presented below was to use a repair approach, but to modify the repair algorithm so that the new operating point is mapped to the point on the simplex closest to the originally generated location. For each point not initially satisfying both constraints, a quadratic programming problem is solved to find the location on the simplex with the smallest squared distance from the generated point ($\bar{w}$):

$$\min_{w \in \mathbb{R}^D} \ \frac{1}{2}\|w - \bar{w}\|_2^2$$

such that

$$0 \leq w_i \leq 1,$$
$$\sum_{i=1}^{k} w_i = 1.$$

Initial testing showed that this constraint enforcement produced better optimization solutions than the other considered constraint-handling methods. The resulting more efficient search procedure makes up for the additional computational load for the quadratic programming step.

4.3 Test Sets and Experimental Results

In this work the MEGA algorithm was compared to the PSO variant, FIPS, and to a standard k-NN which uses uniform weights: $w_i = 1/k$ for all i. The FIPS algorithm was chosen because recent tests showed that it performed extremely well as compared to other PSO implementations. It was implemented with a fully-connected social network, instead of the ring network used for the regional gradient discussion. Both algorithms used $(k+1)^2$ active points following the reasoning from Chapter 2, and were run for $10(k+2)$ function evaluations which allowed MEGA 10 complete rounds of local and global gradient steps. The two global optimization algorithms search for optimal weights satisfying the multi-minima objective function (4.2), with $\lambda = 1$. The k-NN classifier using the uniform weights $w_i = 1/k$. These approaches were tested on the Aural Sonar data set.

The Aural Sonar data set contains similarity data based on returns from a broadband active sonar system. This data set and its original similarity analysis was developed by Philips *et al.* [43]. Signals belong to one of two classes - *target* or *clutter*, and pairwise similarities were assigned to each pair by human subjects. The subjects were unaware of the true class labels and assigned a similarity score from 0 to 5. Each signal was scored by two randomly chosen subjects, and the scores were added to produce a 100×100 matrix with integer values between 0 and 10. Figure 4.1 shows the similarity matrix for this data set. The diagonal represents a sample's self-similarity which was always set to the maximum similarity of 10. Off the diagonal, the image shows the similarity from one sample to another. The first fifty samples are from class one, and the second fifty samples are from class two. One can see a block diagonal structure of the similarity matrix.

Examples of the search space associated with the Aural Sonar data set may be found in Figure 4.2. The six different plots are associated with six different test samples and their 32 nearest neighbors. The axes correspond to the weights W_1 and W_32 on the nearest and furthest neighbors respectively in the $k = 32$ set. In each example, the rest of the dimensions were fixed at $w_i = 1/256$ for $i = 3, \ldots, 31$, and w_2 was set to be $w_2 = 1 - w_1 - \sum_{i=3}^{32} w_i$. It should be noted that these figures show a two-dimensional projection of a thirty-two-dimensional space, so the full thirty-two dimensional problem is more difficult

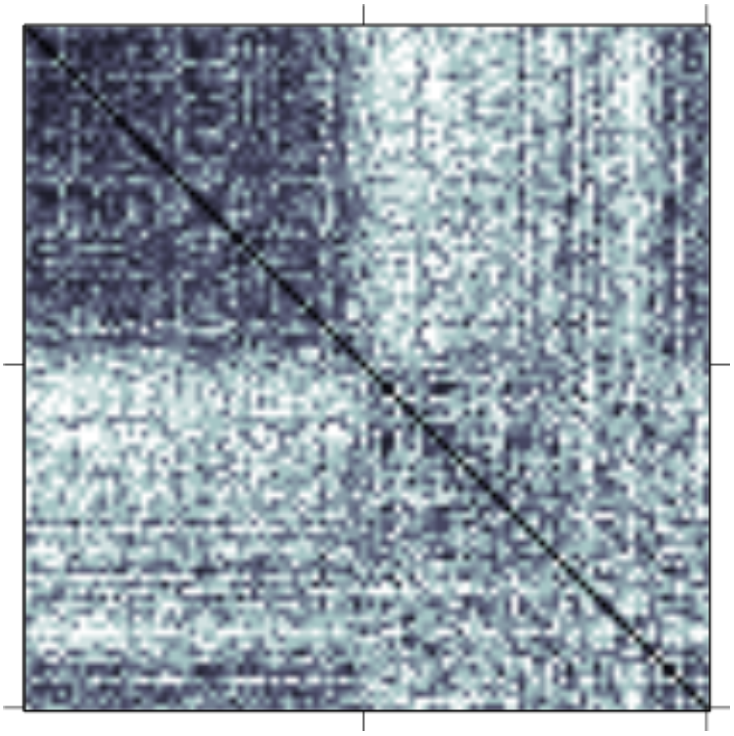

Figure 4.1: The 100×100 similarity matrix for the Aural Sonar data set.

than represented here. These sample search spaces show that, while there is a large amount of structure in each case, the space is often non-convex. This is exactly the sort of problem for which MEGA was designed, because the gradient descent can take advantage of the regular slopes, while the population interaction can avoid getting trapped in saddle points or local minima.

For each test run 20% of the data was withheld for testing, and 80% was used for training. This was repeated with twenty different test-train partitions. Initial comparisons were done with a fixed neighborhood size, k, allowing for a comparison of the objective function solutions as well as the classification errors. Additionally, a second analysis determined the neighborhood size k via a ten-fold cross validation using only the training data, as is standard in the field of statistical learning [19]. k was chosen from the set $\{1, 2, 4, 8, 16, 32\}$. The resulting optimal weights and neighborhood size k were then used to evaluate performance

Table 4.1: Aural Sonar % test data misclassified over 20 randomized training/test partitions for fixed k

	$k = 4$	$k = 8$	$k = 16$	$k = 32$
k-NN	18.25	18.25	18.75	21.50
FIPS *run 1*	16.00	16.50	17.00	17.25
FIPS *run 2*	16.25	17.25	17.75	16.75
FIPS *run 3*	16.50	17.25	18.75	17.25
MEGA-1 *run 1*	14.00	14.75	14.50	14.50
MEGA-1 *run 2*	13.75	14.75	14.50	14.50
MEGA-1 *run 3*	14.50	14.50	14.00	13.75

using the testing data.

The initial question with any optimization solution is whether the search algorithms are performing as expected. In Figure 4.3 each plot shows the objective function solution for a fixed k. The test points are plotted in the same order so the FIPS and MEGA solutions can be directly compared. They are ordered from the lowest MEGA objective value to the highest. It is clear from these plots that MEGA is normally finding a better solution to the objective function than FIPS. A second set of plots in Figure 4.4 shows histograms of the difference between the FIPS and MEGA values for each test point. In these histograms it is easy to see that FIPS is rarely outperforming MEGA, and when it does the margin is small.

The results for the fixed k experiments are shown in Table 4.1. Each row of this table shows the average error for twenty test-train partitions of the data (that is, 400 test samples). The k-NN results are deterministic, but the global optimization algorithms have stochastic elements, so there is some variance in their performances. Each global optimization algorithm has three rows of results associated with it, for three different test runs through to the partitions. This table shows that using FIPS as a global optimizer yields some improvement over the basic k-NN algorithm. Further, using MEGA as the global

Table 4.2: Cross validated k values and associated error rates for Aural Sonar

	mean k	mean % error	std % error
k-NN	6.10	19.50	6.3
FIPS *run 1*	24.4	16.25	7.7
FIPS *run 2*	28.4	15.00	7.5
FIPS *run 3*	28.0	14.50	6.1
MEGA-1 *run 1*	13.3	15.50	6.3
MEGA-1 *run 2*	14.5	15.50	6.7
MEGA-1 *run 3*	12.1	15.00	5.0
MEGA-SD *run 1*	26.8	14.50	6.3
MEGA-SD *run 2*	20.9	13.75	6.9
MEGA-SD *run 3*	26.4	13.00	5.8

optimizer produces the most desirable results by two to three percentage points. In this test MEGA is clearly the superior approach.

In a classification application the choice of k would be cross-validated to ensure that this parameter was properly set. A second test was performed with this method, and the results are in Table 4.2. Each row of these results has the mean and standard deviation of the values for the twenty partitions. Again, the FIPS and MEGA algorithms were run three times to show some of the variability due to their stochastic elements. In this test both the original MEGA (MEGA-1) and the subdivision variant (MEGA-SD) were run. There are three things that are notable in these results. First, the MEGA-SD algorithm is performing better than either FIPS or k-NN. Second, when the choice of k is higher the performance statistics are better. This may imply that when the optimization algorithm is able to choose good weights for more neighbors the performance is improved. It may be noted that the standard deviation of the errors was slightly lower for MEGA than for FIPS. This may be because MEGA is doing a better job converging to a good solution. Third, the results are worse with the cross-validated k choice than they are with any of the fixed k. This often

means that the data is being overfit - that is, a k is chosen that works well with the training samples, but turns out to be suboptimal for the testing samples. Another possibility is that this result is due to the high degree of randomness in the global search algorithms.

4.4 Case Study Discussion

Applied work is the "real" test of an algorithm. As discussed in the beginning of this chapter, MEGA was considered for a variety of applications. For the transmitter localization EM proved more useful, and for a sensor allocation problem PSO was more efficient. Based on experimental evidence, when the objective function had a flat basin such that many x^* resulted in acceptable performance, often both PSO and MEGA found the solution efficiently. With a less computationally expensive objective function, PSO and MEGA both performed well, but MEGA's higher computational load proved to be unacceptable.

In other cases MEGA was able to find better solutions than those found in PSO. In the classification problem presented here both global optimization algorithms obtained better results than the standard k-NN approach. This shows that there is merit in using the indefinite similarity matrix and global optimization to find weights. Further, MEGA showed improvement over the PSO variant FIPS. The search space plots show a structured but non-convex objective function, and it appears that the MEGA algorithm is able to take advantage of this feature to find a better solution than FIPS to the function. From the error analysis it is clear that this improvement in optimization is reflected in the classification problem, as MEGA has the lowest classification errors as well. Additionally, there was no noticeable difference in the computation time of MEGA and FIPS for this application. This is possibly because the quadratic programming that ensures feasibility dominated the computation. These results are encouraging and suggest that as further modifications are made to the MEGA algorithm it may become truly competitive.

One source of difficulty in many applications is a mismatch between the computable objective function and the desired goal. It is possible to find a better solution to the objective function, but not have this improvement reflected in the performance on the real metric. In the classification problem the real metric for performance is the classification results, where the computable objective function is given in (4.2). In this case better solutions to

the objective function are in fact reflected in lower classification errors. In contrast, global optimization was used to localize transmitters in a cognitive radio problem [25]. In this case the objective function measured errors in received power, while the desired metric was errors in (the unknown) transmitters' locations. The difference between the two measures was large enough to marginalize gains from using PSO or MEGA, and another approach was needed. The local optimizer expectation-maximization was used with restart, and proved to be more successful than either global optimizer [39].

Another challenge for MEGA was to perform well enough in the search to compensate for the costlier computational costs. In one case PSO had been used successfully in sensor placement applications [56, 40], but initial attempts to improve those results using MEGA showed only marginal improvement on the objective function metric. In the above static placement problems MEGA's computation time proved to be a liability, but in a similar path planning application, requiring extensive modeling for each function evaluation, preliminary tests showed MEGA to outperform PSO during the same time intervals. In the problem presented in this case study the computational time appears to be limited by the constraint handling, and was acceptable for all the solutions. MEGA is a viable approach for solving similarity-based classification.

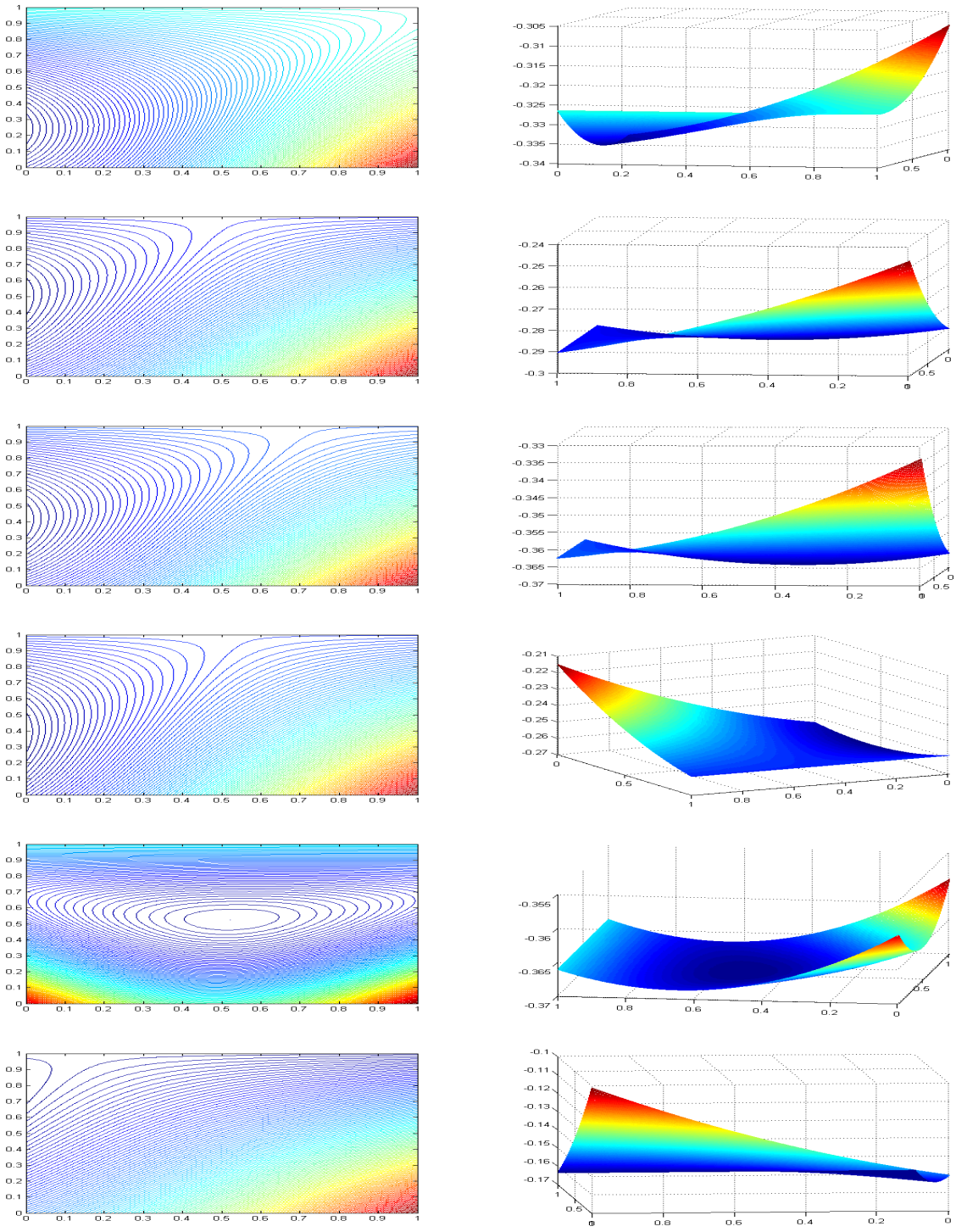

Figure 4.2: 2D projections of the 32D search space for $k = 32$. The axes correspond to the weights W_1 and W_32 on the nearest and furthest neighbors respectively in the $k = 32$ set. The weights $w_3, \ldots, 31 = 1/256$, and $w_2 = 1 - \sum_{i=1,3,\ldots,32} w_i$. The right hand plots are contour plots of the spaces while the left hand plots show sample surfaces.

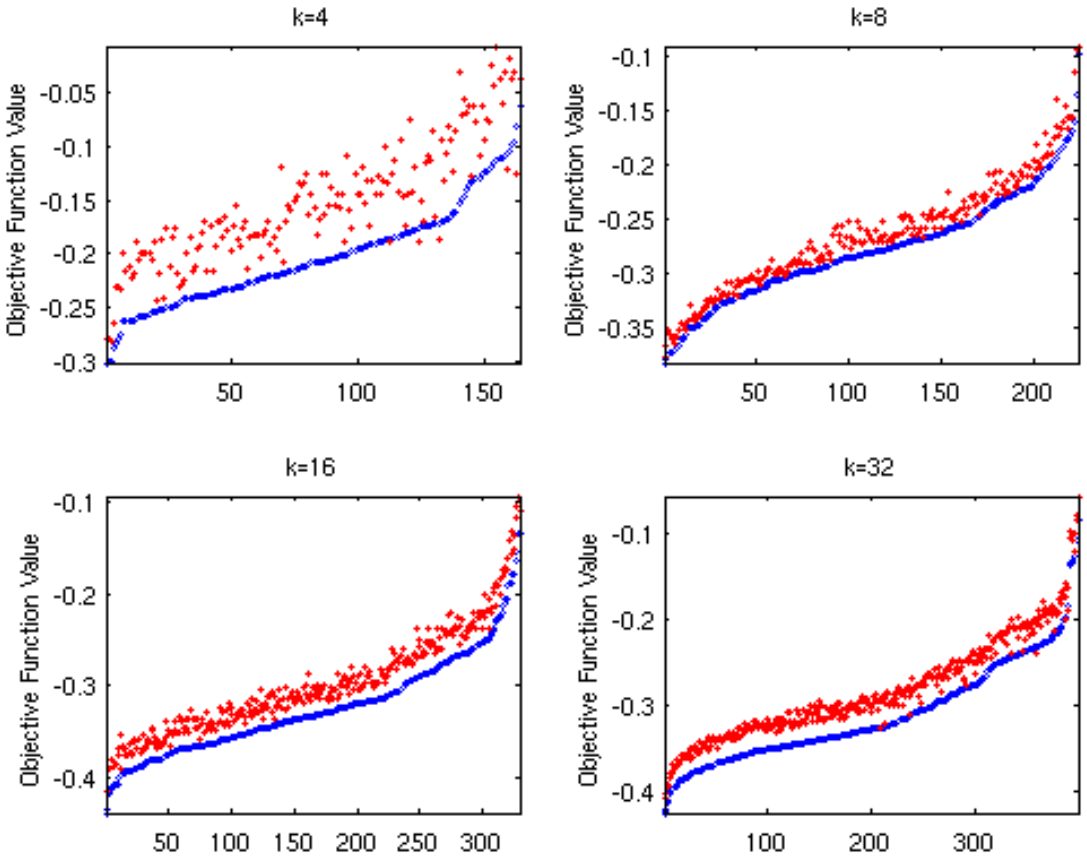

Figure 4.3: Objective function solutions for fixed k tests. The output objective values for the MEGA algorithm are sorted in ascending order and plotted in blue. The FIPS objective values are plotted for the same test point ordering, in red. Two objective function values with the same index number can be directly compared. There are 400 total test samples (20 test samples from each of the 20 randomized partitions). However, if all of a test sample's neighbors are from the same class, then the classification is trivial and the weights are not computed with global optimization. This is more likely to occur for smaller neighborhoods, and so there are fewer samples for the smaller k values.

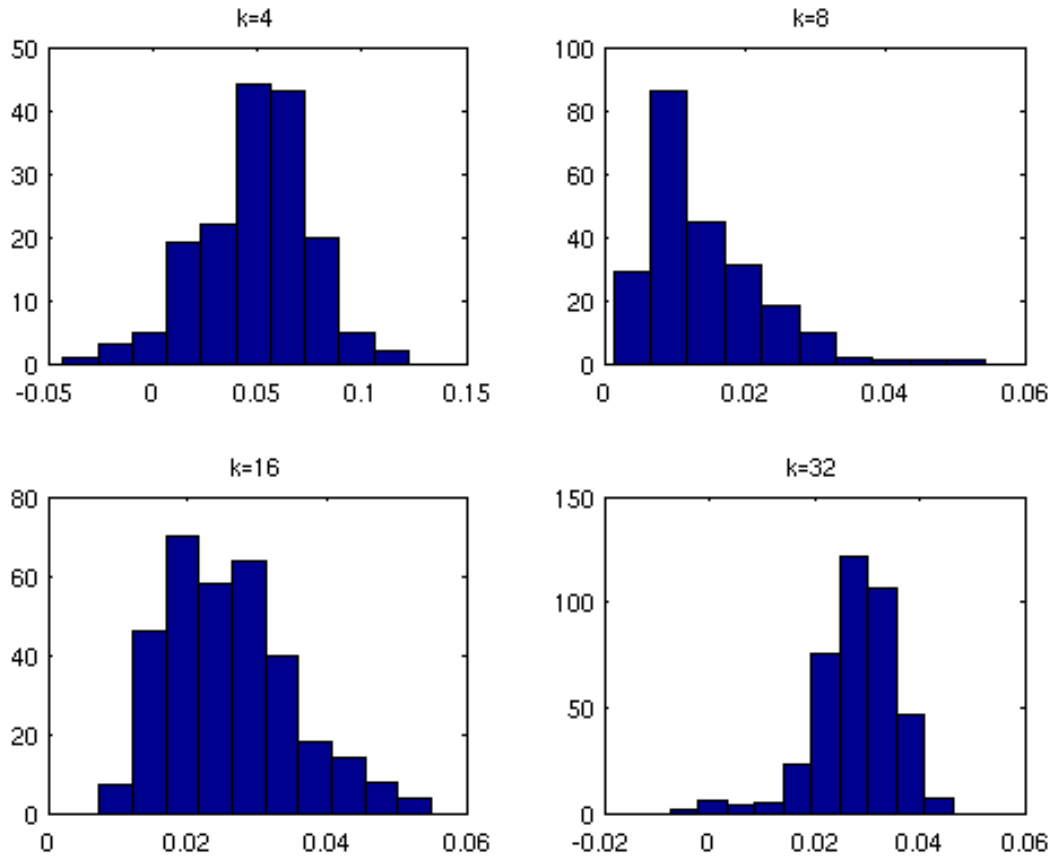

Figure 4.4: Histograms of differences between the FIPS and MEGA objective function solutions. Positive values show the MEGA found a better solution to the objective function than FIPS.

Chapter 5

CONCLUSIONS

This thesis contains an exploration of search strategies for global optimization. The work was driven not by an existing algorithm nor by a single application. Rather, this work was motivated by a desire to understand global optimization from the most basic level. Using a set of fundamental search strategies - gradient descent, multi-resolution searching, and memory - the MEGA algorithm was designed and refined. This algorithm was then used as a platform to further explore each of these strategies. While a number of applications were explored during the course of this research, the problem of similarity-based classification was used as a case study to examine in-depth the performance of the MEGA algorithm. Contributions in this thesis include the proposal of the MEGA algorithm, the proposal of a regional gradient, examination of using the regional gradient in global search, and work on using global optimization to improve the performance of a nearest neighbor classification system.

Chapter 2 presented a few variants of the basic MEGA algorithm, along with a discussion of their pitfalls. If future research can refine one of the proposed variants into a reliable system, or otherwise modify the original algorithm to reduce its computational load, the MEGA algorithm may find a place among the reliable global search strategies used today.

The work presented in Chapter 3 focused on understanding the contribution made different search strategies. In this chapter the regional gradient is proposed and developed. Regional gradient information can play a valuable role in global search. Future work will include using the understanding of regional gradient estimation and population dynamics to improve existing search techniques. PSO, in particular, is a popular and successful algorithm that is well suited for possible improvement through the use of additional function information. Another avenue for improving either PSO or MEGA is to further investigate and refine the population interaction and evolution.

There are open topics for research presented in this thesis. The driving force behind future work will be the solution of engineering problems. Problems in asset allocation, sensor control, and classification offer fertile soil for future optimization application. In Chapter 4 it was demonstrated that the details of a problem can reveal interesting avenues for improvement of basic algorithms. The constraint solution presented here may be improved by a more efficient solution, and different constraints may be better handled with a novel approach. Other problems may reveal different insights into the search strategies themselves.

Chapter 6

APPENDIX: TEST FUNCTION INFORMATION

Branin Problem [11]

$$f(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + g(1-h)cos(x_1) + g \tag{6.1}$$

$$\begin{aligned} a &= 1 \\ b &= \frac{5.1}{4\pi^2} \\ c &= \frac{5}{\pi} \\ d &= 6 \\ g &= 10 \\ h &= \frac{1}{8\pi} \end{aligned}$$

$$-5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15$$

$$x^* = (-\pi, 12.275), (pi, 2.275), (3\pi, 2.475), f(x^*) = \frac{5}{4\pi}$$

Griewank Problem [16]

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

$$-600 \leq x_i \leq 600$$

$$x^* = (0, 0, 0, ...0), f(x^*) = 0$$

Langerman Problem [23]

$$f(x) = -\sum_{j=1}^{5} c_j cos(\pi d_j) exp(\frac{-d_j}{\pi}) \tag{6.2}$$

$$d_j = \sum_{i=1}^{D} (x_i - a_{ji})^2$$

Constants a_{ji} and c_j are available at [23].

$$0 \leq x_i \leq 10$$

$$x^* = (8.074, 8.777, 3.467, 1.867, 6.708, 4.534, 0.276, 7.633, 1.567), f(x^*) = -0.965$$

Quadratic Problem

$$f(x) = \sum_{i=1}^{D} x_i^2$$

$$-1 \leq x \leq 1$$

$$x^* = (0, 0, 0, ..., 0), f(x^*) = 0$$

Rosenbrock Problem [36]

$$f(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

$$-1 \leq x \leq 3$$

$$x^* = (1, 1, 1, ..., 1), f(x^*) = 0$$

Sinusoidal Problem [66]

$$f(x) = -2.5 \prod_{i=1}^{D} sin(x_i - \frac{\pi}{4}) - \prod_{i=1}^{D} sin(5(x_i - \frac{\pi}{4})) + 3.5$$

$$0 \leq x \leq 5$$

$$x^* = (\frac{3\pi}{4}, \frac{3\pi}{4}, \frac{3\pi}{4}, ..., \frac{3\pi}{4}), f(x^*) = 0$$

BIBLIOGRAPHY

[1] http://www.swarmintelligence.org/tutorials.php.

[2] M.M. Ali, C. Khompatraporn, and Z. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31:635–672, 2005.

[3] M.R. AlRashidi and M.E. El-Hawary. Emission-economic dispatch using a novel constraint handling particle swarm optimization strategy. *Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on*, pages 664–669, May 2006.

[4] C. Audet and J.E. Dennis Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2003.

[5] L. Cazzanti and M.R. Gupta. Local similarity discriminant analysis. *Proc. of the 24th International Conference on Machine Learning*, 2007.

[6] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. on Evolutionary Computation*, 6(1):58–72, 2002.

[7] C.A. Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithsm: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, 2002.

[8] A.L. Custódio and L.N. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM J. on Optimization*, 18(2):537–555, 2007.

[9] J.E. Dennis and R.B. Schnabel. *Optimization*, volume 1 of *Handbooks in OR and MS*, chapter : A view of unconstrained optimization, pages 1–72. Elsevier, New York, 1989.

[10] L.P. Devroye. On the convergence of statistical search. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-6(1):46–56, January 1976.

[11] L. Dixon and G. Szegó. *Towards Global Optimization*, volume 2. North Holland, New York, 1978.

[12] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, January 2002.

[13] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. on Systems, Man, and Cybernetics*, 26(1):1–13, 1996.

[14] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. IEEE, October 1995.

[15] F. Glover and M. Laguna. *Handbook of Applied Optimization*, chapter 3.6.7: Tabu Search, pages 194–208. Oxford University Press, New York, NY, 2002.

[16] A.O. Griewank. Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, 34:11–39, 1981.

[17] M.R. Gupta, R.M. Gray, and R.A. Olshen. Nonparametric supervised learning by linear interpolation with maximum entropy. *IEEE Trans. on Pattern Anal. and Machine Intel.*, 28(5):766–781, May 2006.

[18] B. Hannaford. Resolution-first scanning of multidimensional spaces. *Graphical Models and Image Processing*, 55(5):359–369, September 1993.

[19] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer-Verlag, New York, 2001.

[20] M. Hazen and M.R. Gupta. A multiresolutional estimated gradient architecture for global optimization. pages 3013–3020. IEEE Congress on Evolutionary Computation, July 2006.

[21] A. E. Hoerl and R. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

[22] R. Hooke and T.A. Jeeves. Direct search solution of numerical and statistical problems. *J.ACM*, 8:212–229, 1961.

[23] http://iridia.ulb.ac.be/ aroli/ICEO/2ndICEO.html. Internet Publication.

[24] Xiaohui Hu, R.C. Eberhart, and Yuhui Shi. Engineering optimization with particle swarm. *Proc. of the 2003 IEEE Swarm Intelligence Symposium, 2003. SIS '03.*, pages 53–57, April 2003.

[25] M.R. Gupta J.K. Nelson, M.U. Hazen. Global optimization for multiple transmitter localization. *Military Communications Conference, 2006. MILCOM 2006*, pages 1–7, Oct. 2006.

[26] D. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[27] C. Khompatraporn, J. Pinter, and Z. Zabinsky. Comparative assessment of algorithms and software for global optimization. *Journal of Global Optimization*, 31:613–633, 2005.

[28] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science, 13 May 1983*, 220(4598):671–680, 1983.

[29] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[30] M. Locatelli. On the multilevel structure of global optimization problems. *Computational Optimization and Applications*, 30:5–22, 2005.

[31] D.G. Luenberger. *Linear and nonlinear programming.* Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 1989.

[32] J.L. Maryak and D.C. Chin. Global random optimization by simultaneous perturbation stochastic approximation. In B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, editors, *Proc. of the 2001 Winter Simulation Conference*, pages 307–312, 2001.

[33] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: simpler, maybe better. *IEEE Trans. on Evolutionary Computation*, 8(3):204–210, 2004.

[34] B. Meyer. Constraint handling and stochastic ranking in aco. *The 2005 IEEE Congress on Evolutionary Computation, 2005.*, 3:2683–2690 Vol. 3, Sept. 2005.

[35] L. Mišík, J. Tvrdík, and I. Křivý. On convergence of a class of stochastic algorithms. In J. Antoch and G. Dohnal, editors, *Proc. of ROBUST 2000*, pages 198–209, Prague, 2001. JČMF Press.

[36] J. Moré, B. Garbow, and K. Hillstrom. Testing unconstrained optimization software. *ACM Trans. on Mathematical Software*, 7:17–41, 1981.

[37] R.H. Myers and D.C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiements.* Wiley-Interscience, USA, 2002.

[38] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–311, 1965.

[39] J.K. Nelson and M.R. Gupta. An em technique for multiple transmitter localization. *41st Conference on Information Science and Systems*, 2007.

[40] P.N. Ngatchou, W.L.J. Fox, and M.A. El-Sharkawi. Distributed sensor placement with sequential particle swarm optimization. *Proc. of IEEE Swarm Intelligence Symposium*, pages 385 – 388, 2005.

[41] M.M. Noel and T.C. Jannett. Simulation of a new hybrid particle swarm optimization algorithm. *Proc. of the Thirty-Sixth Southeastern Symposium on System Theory, 2004.*, pages 150–153, 2004.

[42] P.Y. Papalambros and D.J. Wilde. *Principles of Optimal Design*. Cambridge University Press, Cambridge, UK, 1988.

[43] S. Philips, J. Pitton, and L. Atlas. Perceptual feature identification for active sonar echoes. In *Proc. of the 2006 IEEE OCEANS Conf*, 2006.

[44] D.A. Pierre. *Optimization Theory With Applications*. Dover Publications, Inc., New York, NY, 1986.

[45] O. Polgar, M. Fried, and I. Barsony. A combined topographical search strategy with ellipsometric application. *Journal of Global Optimization*, 19:383–401, 2001.

[46] M.J.D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336, 1998.

[47] W.L. Price. A controlled random search procedure for global optimization. *The Computer Journal*, 20(4):367–370, 1976.

[48] T.P. Runarsson and Xin Yao. Search biases in constrained evolutionary optimization. *Proc. of the Thirty-Sixth Southeastern Symposium on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(2):233–243, May 2005.

[49] J.F. Schutte and A.A. Groenwold. A study of global optimization using particle swarm. *Journal of Global Optimization*, 31:93–108, 2005.

[50] O.S. Soliman, L.T. Bui, and H.A. Abbass. The effect of a stochastic step length on the performance of the differential evolution algorithm. *IEEE Congress on Evolutionary Computation, 2007. CEC 2007.*, pages 2850–2857, Sept. 2007.

[51] D. Solomatine. Two strategies of adaptive cluster covering with descent and their comparison to other algorithms. *Journal of Global Optimization*, 14:55–79, 1999.

[52] J.C. Spall. *Introduction to Stochastic Search and Optimization*. Wiley-Interscience, Hoboken, NJ, 2003.

[53] J.C. Spall, S.D. Hill, and D.R. Stark. *Theoretical Framework for Comparing Several Stochastic Optimization Approaches*, chapter 3. Springer, Berlin, 2006.

[54] C.P. Stephens and W. Baritompa. Global optimization requires global information. *Journal of Optimization Theory and Applications*, 96(3):575–588, March 1998. sets limitations on global search.

[55] T. Takahama and S. Sakai. Solving constrained optimization problems by the constrained particle swarm optimizer with adaptive velocity limit control. *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–7, June 2006.

[56] B. Thompson. *Inversion and Fast Restoration Using Computational Intelligence with Applications to Geoacoustics*. PhD thesis, University of Washington, Department of Electrical Engineering, 2004.

[57] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[58] A. Törn, M.M. Ali, and S. Vitanen. Stochastic global optimization: Problem classes and solution techniques. *Journal of Global Optimization*, 14:437–447, 1999.

[59] F. van den Bergh and A.P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Trans. on Evolutionary Computation*, 8(3):225–239, June 2004.

[60] S. Venkatraman and G.G. Yen. A generic framework for constrained optimization using genetic algorithms. *IEEE Trans. on Evolutionary Computation*, 9(4):424–435, Aug. 2005.

[61] Xiao-Feng Xie, Wen-Jun Zhang, and De-Chun Bi. Handling equality constraints by adaptive relaxing rule for swarm algorithms. *Congress on Evolutionary Computation, 2004. CEC2004.*, 2:2012–2016 Vol.2, June 2004.

[62] X. Yao, Y. Liu, and G.M. Lin. Evolutionary programming made faster. *IEEE Trans. on Evolutionary Programming*, 3:82–102, July 1999.

[63] K.F.C. Yiu, Y. Liu, and K.L. Teo. A hybrid descent method for global optimization. *Journal of Global Optimization*, 28:229–238, 2004.

[64] Z.B. Zabinsky. Stochastic methods for practical global optimization. *Journal of Global Optimization*, 13:433–444, 1998.

[65] Z.B. Zabinsky. *Stochastic Adaptive Search for Global Optimization*. Kluwer Academic Publishers, Norwell, MA, 2003.

[66] Z.B. Zabinsky, D.L. Graesser, M.E. Tuttle, and G.I. Kim. *Recent Advances in Global Optimization*, chapter : Global optimization of composite laminates using improving hit and run, pages 343–368. Princeton University Press, 1992.

[67] Wen-Jun Zhang, Xiao-Feng Xie, and De-Chun Bi. Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. *Congress on Evolutionary Computation, 2004. CEC2004.*, 2:2307–2311 Vol.2, June 2004.

VITA

Megan Hazen was born in Pittsburgh, Pennsylvania, and grew up in Pennsylvania and Connecticut. Ms. Hazen attended Carnegie Mellon University, earning a Bachelors of Science (1997) and a Masters of Science (1999) in the Mechanical Engineering program. At the University of Washington she earned a Masters of Science (2004) and a Doctor of Philosoopy (2008) in the Electrical Engineering program. Ms. Hazen's graduate work in both mechanical and electrical engineering has focused on the use of intelligent systems, including global optimization, for engineering problems. Since moving to Seattle in 2000, Ms. Hazen has worked for the Applied Physics Laboratory at the University of Washington where her work has centered around the use of intelligent systems in sonar applications.