AN ADAPTIVE RANDOM SEARCH ALGORITHM WITH LINEAR COMPLEXITY IN DIMENSION

Zelda B. Zabinsky
Industrial Engineering Program, FU-20
University of Washington
Seattle, Washington 98195

Robert L. Smith
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 90-15

April 27, 1990

AN ADAPTIVE RANDOM SEARCH ALGORITHM WITH LINEAR COMPLEXITY IN DIMENSION

Zelda B. Zabinsky †
Industrial Engineering Program, FU-20
University of Washington
Seattle, Washington 98195

Robert L. Smith
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109

April 27, 1990

Abstract

Random search algorithms have been developed to solve global optimization problems with many local minima. Such methods are only attractive if their computational effort does not grow exponentially with dimension. In this paper, a sequential random search algorithm is analyzed on a restricted subset of convex problems and shown to have linear complexity in dimension.

Key Words: Random search, Monte Carlo optimization, algorithm complexity.

† The work of Zelda B. Zabinsky was partially supported by the Graduate School Research Fund at the University of Washington.

1 Introduction

Random search techniques show promise as efficient optimization methods for a large class of problems. Recent research [13,23] shows that it is theoretically possible for a random search algorithm to achieve a complexity that is, on the average, linear in dimension. In this paper, we define and implement a sequential random search algorithm named the Adaptive Mixing Algorithm (AMA). We analyze the computational efficiency of the AMA for a restricted subset of convex programs, and use theoretical and computational results to show that an upper bound on the number of iterations is linear in dimension.

Sequential random search procedures are designed to address a standard optimization problem,

$$(P) \qquad \min_{x \in S} f(x)$$

where x is an n-dimensional vector, S is a convex, compact subset of $\mathbf{R}^n$, and f is a real-valued continuous function defined over S. All sequential random search procedures generate a sequence of random points $\{X_k\}$ which may depend on the previous point or several of the previous points. The concept underlying sequential step size algorithms [14,15,12,19,17,9,18] is to generate the next random point X_{k+1} by taking a specified step in a random direction from the previous point X_k. These algorithms are based on the iterative formula, for $k = 1, 2, \ldots$

$$X_{k+1} = \begin{cases} X_k + s_k D_k & \text{if } f(X_k + s_k D_k) < f(X_k) \\ X_k & \text{otherwise} \end{cases}$$

where D_k is the direction vector, s_k is the step size, and X_k is the point generated in the k^{th} iteration. The direction vector is usually obtained by sampling from a uniform distribution on a unit hypersphere. The method of choosing the step size is specific to each algorithm.

Several of the step size algorithms have been reported in the literature to have complexity that increases linearly in dimension. Complexity is measured as the number of function evaluations needed to reach a specified neighborhood of the solution. Three key references are:

(a) Schumer and Steiglitz [19] prove that the average number of function evaluations for an optimum relative step size random search restricted to an unconstrained hyperspherical objective function is asymptotically linear in dimension. They also provide experimental evidence that the number of function evaluations increases linearly with dimension for their adaptive step size algorithm on the following three test functions;

$$\sum_{i=1}^{n} {x_i}^2, \qquad \sum_{i=1}^{n} {x_i}^4, \qquad \sum_{i=1}^{n} a_i {x_i}^2.$$

(b) Schrack and Borowski [17] report experimental results that doubling the dimension doubles the number of function evaluations required for another random search algorithm. They also used a hyperspherical test function, $\sum_{i=1}^{n} x_i^2$.

(c) Solis and Wets [21] experimentally verified a linear correlation between the number of function evaluations and dimension for their own variation of the step size algorithm on a hyperspherical test function. They provided a justification of this linearity condition based on the tendency of these algorithms to maintain a constant probability of successful improvement.

All of these findings empirically suggest that sequential random search procedures are appropriate for large dimensional quadratic programs. An analysis of a random search procedure called pure adaptive search [13,23] substantiates these findings, and proves that it is theoretically possible for a random search procedure to achieve linear complexity. In Zabinsky and Smith [23], the linear complexity, measured as the expected number of iterations to get arbitrarily close to the solution, was generalized to global optimization problems. Pure adaptive search constructs a sequence of points uniformly distributed within a corresponding sequence of nested improving regions. As pointed out in [13,23], pure adaptive search is difficult to implement directly due to the problem of efficiently generating a point according to a uniform distribution in a general region. We seek an algorithm that can be implemented directly, with a theoretical analysis that the number of iterations is again linear in dimension.

2 Adaptive Mixing Algorithm

The Adaptive Mixing Algorithm is intended to be easy to implement and to approximate the efficiency of pure adaptive search. The question is: how can an improving point be easily generated that randomly samples the improving level set? One answer to this question is to use Hit-and-Run methods [20,1,2,7]. Hit-and-Run generates a sequence of random points by providing a random direction and then providing a random point in that direction. This sequence of points asymptotically approach a uniform distribution [20]. Thus, an implementation of pure adaptive search is to use a sequence of Hit-and-Run points on each iteration. Now a decision must be made regarding the best length of the Hit-and-Run sequence per iteration. In fact, a class of algorithms can be parametrically defined by the length of the Hit-and-Run sequences. In one extreme, we know that when the Hit-and-Run sequences are very long and provide a close approximation to sampling from a uniform distribution we have a good approximation to pure adaptive search and can expect the number of iterations to be linear in dimension. In another extreme, consider an algorithm where the Hit-and-Run sequences are very short - a length of *one*. Now what is the expected number of iterations? We will call the algorithm with Hit-and-Run sequences of length one, the Adaptive Mixing Algorithm. The analysis will express the number of iterations as a

function of dimension.

The Adaptive Mixing Algorithm also fits into the framework of a sequential random search procedure. Given an initial feasible point, X_0, AMA generates a random direction vector that originates at X_0, and then generates an improving and feasible point X_1 uniformly distributed along the line segment associated with that direction. Thus, the next point is in a random direction and a random distance from the previous point. The procedure proceeds iteratively to produce a sequence of improving points until satisfying a stopping criterion.

More formally,

Adaptive Mixing Algorithm (AMA)

Step 0. Initialize $X_0 \in S$, $Y_0 = f(X_0)$, and $k = 0$.

Step 1. Generate a direction vector D_k uniformly distributed over a unit hypersphere.

Step 2. Generate X_{k+1} uniformly on L_k, the *improving* feasible line segments in direction D_k, i.e. $L_k = \{x : x = X_k + \lambda D_k,\ \lambda \in \mathbf{R},\ f(x) < f(X_k) \text{ and } x \in S\}$. Set $Y_{k+1} = f(X_{k+1})$. If $L_k = \emptyset$, go to Step 1.

Step 3. If the stopping criterion is met, stop. Otherwise increment k and return to Step 1.

Details of the implementation of the AMA are given in the appendix. Generating random directions, as defined in Step 1, is straightforward. However, implementing Step 2 is more involved, and we experimented with several approximating forms. Our implementation tries to find a random point on the improving feasible line segment, L_k. If it fails after several tries, it returns to Step 1 to generate another random direction. The computer program stops if no improving point is found after a user-defined maximum number of tries.

3 Complexity Analysis

3.1 Definitions and Notation

Before proceeding with the analysis, we define a complexity measure for evaluating AMA.

For the optimization problem (P), let (x_*, y_*) denote the solution, where

$$x_* = \arg\min_{x \in S} f(x)$$

and

$$y_* = f(x_*) = \min_{x \in S} f(x).$$

The value x_* need not be unique; any point $x_* \in S$ that attains the minimum, $f(x_*) = y_*$, may be used for a solution point. It will also be convenient to define the maximum,

$$y^* = \max_{x \in S} f(x).$$

Let P_y denote the *level set* of the problem (P) at objective function value y,

$$P_y = \{x : f(x) < y, x \in S\}$$

for $y_* < y < y^*$. Let $\{X_k\}$, $X_k \in S$, $k = 0, 1, 2, \ldots$ be the sequence of *points* generated by the Adaptive Mixing Algorithm on (P), with the subscript k denoting iteration count. Let $\{Y_k\}$, $y_* \leq Y_k \leq y^*$, $k = 0, 1, 2, \ldots$ be the corresponding sequence of *objective function values* generated by the algorithm on (P), $Y_k = f(X_k)$.

The sequences $\{X_k\}$ and $\{Y_k\}$ are random variables due to the stochastic nature of a Monte Carlo algorithm. The *distribution of improvement* is defined to be the probability that the k^{th} objective function value Y_k is at most y. This is the probability that the k^{th} point lies within the level set P_y, so that

$$P(Y_k \leq y) = P(X_k \in P_y)$$

for $k = 0, 1, 2, \ldots$ and $y_* \leq y \leq y^*$. The *conditional distribution of improvement*, denoted,

$$P(Y_{k+1} \leq y | Y_k = w)$$

for $k = 0, 1, 2 \ldots$ and $y_* \leq y \leq y^*$ is defined to be the probability of obtaining an objective function value of at most y in a single iteration, starting from objective function value w. Let $\{Z_k\}$, $0 \leq Z_k \leq 1$, $k = 0, 1, 2, \ldots$ be the sequence of *normalized improvements* associated with the points generated by the algorithm on (P), where

$$Z_k = (Y_k - y_*)/(y^* - y_*).$$

The probability of achieving a normalized improvement of z by the k^{th} iteration is,

$$P(Z_k \leq z) = P(Y_k \leq y)$$

for $0 \leq z \leq 1$ and $z = (y - y_*)/(y^* - y_*)$.

We say an algorithm has achieved *m-fold improvement* on (P) when the normalized improvement is within $1/m$ of the optimal value, for $m > 1$:

$$0 \leq Z_k \leq 1/m$$

or equivalently, when

$$y_* \leq Y_k \leq y_* + (1/m)(y^* - y_*).$$

As a measure of computational complexity, an *iteration count*, $K_{\alpha,m}$, is defined as the smallest number of iterations needed to achieve m-fold improvement with $1-\alpha$ certainty,

$$K_{\alpha,m} = \min_{k=1,2,\ldots} \{k : P(Z_k \leq 1/m) \geq 1-\alpha\}$$

for $m > 1$ and $0 < \alpha < 1$.

This iteration count, $K_{\alpha,m}$, will be used as a measure of computational complexity throughout the paper. It is the same complexity measure used in [13]. Other measures of complexity count the number of function evaluations, gradient evaluations, or arithmetic operations, whereas $K_{\alpha,m}$ counts the number of iterations. The use of normalized improvement in $K_{\alpha,m}$ is similar to the error measure of an upper bound used in [3]. They also scale a deviation from the optimum by a range on objective function values. A contrasting complexity measure is the rate of convergence, which characterizes the tail of the sequence converging to the solution [10]. Instead of measuring the tail, $K_{\alpha,m}$ measures the number of iterations needed to get arbitrarily close to the solution.

3.2 Theoretical Analysis

We now turn to an analysis of the complexity of the Adaptive Mixing Algorithm. For the analysis, we define $Z_0 = 1$ and $Y_0 = y^*$ as initial starting conditions.

To begin the analysis, we must specify the conditional distribution of improvement for the Adaptive Mixing Algorithm. The conditional probability of making a specified improvement on a single iteration depends on the position of the current point,

$$\begin{aligned} P(Y_{k+1} \leq y | Y_k = w) &= E[P(Y_{k+1} \leq y | X_k, Y_k = w)] \\ &= E[P(X_{k+1} \in P_y | X_k, Y_k = w)]. \end{aligned}$$

This probability can be expressed in terms of the random direction generated from the point x, and the ratio of the length of improving line segment(s) in that direction,

$$P(X_{k+1} \in P_y | X_k = x, Y_k = w) = \int_d \|L_{P_y}(d,x)\| / \|L_{P_w}(d,x)\| dF_D(d)$$

where $\|L_{P_y}(d,x)\|$ is the combined length of the line segments formed by the intersection of the level set P_y with the direction vector d originating at x. Also, $F_D(\cdot)$ is the cumulative distribution function for the random direction vector. In words, the probability of making an improvement to y, starting at point x with $f(x) = w$, is the probability of landing within P_y given direction d, i.e. $\|L_{P_y}(d,x)\| / \|L_{P_w}(d,x)\|$, integrated over all feasible improving directions.

For general convex programs, the conditional probability of improvement depends on the exact location of X_k, and makes it difficult to derive a general expression. However, for a

subset of convex programs with symmetrical level sets, we can continue the analysis. This class of programs includes all the test functions for which linearity has been cited [19,17,21].

Consider the class of quadratic programs with "spherical" level sets, that is, unconstrained quadratic programs with a constant diagonal Hessian (cI, where $c \geq 0$ and I is the identity matrix), which we will refer to as *spherical quadratic programs.* With spherical quadratic programs, the symmetry of the level sets ensures that the conditional probability of improvement is invariant for all points on a level set;

$$P(X_{k+1} \in P_y | X_k = x, Y_k = w) = P(X_{k+1} \in P_y | X_k = x', Y_k = w)$$

for any x and x' with $f(x) = f(x') = w$. The conditional probability of improvement for the Adaptive Mixing Algorithm on spherical quadratic programs is:

$$\begin{aligned} P(Y_{k+1} \leq y | Y_k = w) &= E[P(X_{k+1} \in P_y | X_k, Y_k = w)] \\ &= P(X_{k+1} \in P_y | X_k = x, Y_k = w) \qquad \text{for } \textit{any } x \text{ with } f(x) = w, \\ &= \int_d \|L_{P_y}(d,x)\| / \|L_{P_w}(d,x)\| dF_D(d). \end{aligned}$$

Thus, the conditional probability of improvement depends on the ratio of the lengths of line segments and the objective function value (w), but not on the specific point x.

Of course, this class of spherical quadratic programs may be trivial to solve with conventional methods, but the Adaptive Mixing Algorithm is not meant to compete with quadratic programming algorithms. However, if a spherical quadratic program was perturbed slightly, due to noisy data for example, or perhaps a sinusoidal component, it may be very difficult to solve with conventional approaches. On the other hand, we would expect the performance of AMA to be insensitive to perturbations of this type.

Finally, to obtain an upper bound on AMA's iteration count $K_{\alpha,m}$, we define a worst case problem. For any spherical quadratic program (P), define (Q) to be its corresponding *conical program,*

$$(Q) \qquad \min_{x \in S} g(x)$$

where $g(x) = \inf\{y : (x,y) \in C\}$, and C = convex hull of (x_*, y_*) and (S, y^*). This conical program is a worst case problem, in the sense that $K_{\alpha,m}$ on (Q) is greater than or equal to $K_{\alpha,m}$ on (P). The proof is included in the appendix (see also [13]).

Now, let μ be the *expected ratio of normalized improvement* on the conical program,

$$\mu = E\left[Z^Q_{k+1} / Z^Q_k\right].$$

For the AMA on the class of conical spherical quadratic programs, μ does not depend on the value of k. In fact, Z^Q_{k+1}/Z^Q_k is independent and identically distributed for all $k = 0, 1, 2, \ldots$

(refer to Lemma 3.0 in the appendix). However, μ is a function of the dimension of the problem and may be written,

$$\mu(n) = E\left[Z_{k+1}^{Q}/Z_{k}^{Q}\right].$$

We are now ready to state an analytical bound on complexity for the AMA.

Theorem 3.1 *Given the Adaptive Mixing Algorithm and any spherical quadratic program* (P) *with* $\mu(n)$ *for its conical program* (Q)*, then for any* $m > 1$ *and* $0 < \alpha < 1$*,*

$$K_{\alpha,m} \leq \frac{2\ln\left(m(1+\alpha^{-1/2})\right)}{\ln(1/\mu(n))}.$$

Proof: See Appendix. ∎

Theorem 3.1 provides an upper bound on the iteration count as a function of dimension (n), the amount of improvement (m), and the degree of certainty (α). To see how efficient the Adaptive Mixing Algorithm is in terms of dimension, we need to hold m and α constant, and express μ in terms of dimension. Unfortunately, $\mu(n)$ is difficult to express analytically. However, we can experimentally approximate it as a function of dimension.

3.3 Parameter Estimation

The parameter $\mu(n) = E\left[Z_{k+1}^{Q}/Z_{k}^{Q}\right]$ was estimated by collecting normalized improvements Z_k from 2 runs of 100 iterations each on a conical program. Thus, the estimates $\hat{\mu}(n)$ were based on 200 ratios at each dimension, $n = 2, 4, 6, 8, 10, 20, 30, 40$, and 50. The estimates are presented in Table 1 (analytically $\mu(1) = 0.5$). The experiments were run on the following conical program,

$$(Q.1) \qquad \begin{array}{lll} \text{minimize} & 10\left(\sum_{i=1}^{n}(x_i - 5)^2\right)^{1/2} & \\ \text{subject to} & 0 \leq x_i \leq 10 & \text{for } i = 1, 2, \ldots, n \end{array}$$

Notice that the test function $(Q.1)$ is a conical program corresponding to a hyperspherical test function, such as $(P.1)$ defined later.

Now the estimates $\hat{\mu}(n)$ will be used to establish the effect of dimension on the bound on iteration count. As stated in Theorem 3.1,

$$K_{\alpha,m} \leq 2\left(\frac{1}{\ln(1/\mu(n))}\right)\ln\left(m(1+\alpha^{-1/2})\right)$$

so the only term with dimension is $1/\ln(1/\mu(n))$. Figure 1 plots this function of the estimates,

$$1/\ln(1/\hat{\mu}(n))$$

versus dimension. There is one point per run. A linear model was fit to the experimental data, yielding

$$1/\ln(1/\hat{\mu}(n)) = (3.5 \pm 0.2)n + (3.2 \pm 4.8).$$

Adding a quadratic term to the above regression did not significantly improve the fit. The quadratic model fit to the experimental data was,

$$1/\ln(1/\hat{\mu}(n)) = (-0.02 \pm 0.02)n^2 + (4.5 \pm 0.8)n + (3.2 \pm 6.7).$$

Using the coefficients from the linear model, we conclude that the bound on iteration count can be well approximated by a *linear* function of dimension,

$$K_{\alpha,m} \leq 2\,(3.5n + 3.2) \ln\left(m(1 + \alpha^{-1/2})\right).$$

As an example, Table 2 provides this estimated bound on iteration count, $K_{\alpha,m}$, for spherical quadratic programs in various dimensions, with a million-fold improvement ($m = 10^6$) and 99% certainty ($\alpha = 0.01$). The numbers were calculated as the next greatest integer of $2(3.5n + 3.2)\ln(m(1 + \alpha^{-1/2})$. Similar calculations were made to graph the bound on $K_{\alpha,m}$ in Figures 2 and 3.

Thus far, the analysis has concentrated on evaluating the effect of dimension on the iteration count while keeping m and α constant. It is also interesting to note the effect of the accuracy of solution (m and α) on the iteration count. The m-fold improvement and $1 - \alpha$ degree of certainty are both represented logarithmically in the upper bound on iteration count. This might explain the experience for the AMA and other algorithms that large improvements are made very efficiently, while refinements of the solution become costly.

4 Computational Results

4.1 Computational Experiments

We report two computational experiments to observe the effect of dimension on computation and to give a rough comparison to previous algorithms [17,19,21].

The first experiment was on the "conical" convex program ($Q.1$). This function was chosen because it is a worst case problem for AMA, as previously discussed. The Adaptive Mixing Algorithm was run using ten random seeds at each dimension, from $n = 2$ to $n = 50$, to compute an average number of iterations. The experiment used a stopping criterion of 100-fold improvement, and a starting position on a side of the feasible region, $(5, 5, \ldots, 5, 10)$. The results are shown in Figure 2, where the average number of iterations are graphed as a function of dimension. The theoretical upper bound on complexity (as derived in the Section 3) is indicated by the heavy line. This bound is for 50% certainty of 100-fold

improvement ($K_{\alpha,m}$ with $m = 100$ and $\alpha = 0.5$), while the data shows 100% certainty of 100-fold improvement. All of the observed points satisfy the bound. It is clear from this experiment that the upper bound is not tight. The light line on the graph represents a linear regression on the mean number of iterations. The regression coefficients were $27n - 83$ with r^2 of 0.994.

The second experiment was on a hyperspherical problem. This function was chosen because it was the only test function common to [17,19,21] that had been varied by dimension. The Adaptive Mixing Algorithm was run on the following hyperspherical problem,

$$(P.1) \qquad \begin{array}{ll} \text{minimize} & \sum_{i=1}^{n}(x_i)^2 \\ \text{subject to} & -10 \leq x_i \leq 10 \quad \text{for } i = 1, 2, \ldots, n \end{array}$$

with a starting point on a side of the feasible region, $(10, 0, \ldots, 0)$. Five random seeds were used at each dimension, from $n = 2$ to $n = 40$. The experiment used a stopping criterion of $1,000$-fold improvement. The graph in Figure 3 provides the average number of iterations as a function of dimension. The upper bound on complexity is for 50% certainty of $1,000$-fold improvement ($K_{\alpha,m}$ with $m = 10^3$ and $\alpha = 0.5$) and is indicated by the heavy line in the figure. The data of $1,000$-fold improvement with 100% certainty again satisfies the bound. Again, the light line on the graph is the linear regression line on the mean number of iterations, estimated as $11n - 15$ with r^2 of 0.993. To summarize, the experiments indicate that the complexity of the AMA is well approximated by a linear function in dimension.

4.2 Comparisons Between Algorithms

A direct comparison between the AMA and other algorithms is difficult for a number of reasons. The first difficulty in comparing these algorithms is that they were tested on different problems. Although the conical program gives a worst case and hence upper bound on complexity, we also tested the hyperspherical test function to be consistent with [17,19,21].

A second difficulty in comparing the performance of these algorithms is due to the dependency on the accuracy of solution. Our complexity analysis shows an exact dependency on number of iterations to achieve an m-fold improvement. Because of this predicted dependency, the AMA used 10^3-fold improvement as a stopping criterion on the hyperspherical test function to be consistent with Solis and Wets [21]. Schumer and Steiglitz [19] used an $\epsilon = 10^{-8}$ as a stopping criterion. This is only approximately 10^8-fold improvement due to the various starting positions used. Schrack and Borowski [17] used the rate of average convergence as a comparison, instead of a specified level of improvement.

A third difficulty in comparing algorithms across dimension is that they were all tested for varying dimensions. The AMA and Schumer and Steiglitz's algorithm were tested on dimensions up to 40 on the hyperspherical function, while Solis and Wets' algorithm was tested up to 10 dimensions. Schrack and Borowski only compared $n = 5$ with $n = 10$.

The fourth difficulty in comparing performance was that [19,21] both reported average number of function evaluations, while the analysis on the AMA focused on the number of iterations. For comparison purposes, the average number of function evaluations for the AMA on the hyperspherical test function, to achieve 10^3-fold improvement for dimensions up to 40 was approximately $137n$. Schumer and Steiglitz [19] reported the average number of function evaluations of their Adaptive Step Size Algorithm on a hyperspherical problem to get within $\epsilon = 10^{-8}$ of the solution for dimensions up to 40 as approximated by $80n$. Solis and Wets [21] tested their version of an adaptive step size algorithm on the same hyperspherical problem up to 10 dimensions, and reported that the average number of function evaluations to achieve 10^3-fold improvement as roughly $33n$. Schrack and Borowski [17] did not attempt to quantify the effect of dimension, but reported "for a given reduction of the function evaluations, a doubling of dimension doubles the number of function evaluations".

The attraction to the AMA is that it is relatively simple to implement, and has potential for more efficient implementations. The focus in this paper and in the development of the algorithm was aimed at iterations, rather than function evaluations. This gives some support as to the value of short Hit-and-Run sequences within an optimizing framework. Also, to compare with the linearity results reported, AMA has a linear bound on complexity in terms of iterations for a broader class of functions than just the hyperspherical function.

5 Summary

The Adaptive Mixing Algorithm has been shown to have a search effort that is nearly linear in dimension for the class of spherical quadratic programs. The functional bound on complexity also includes the accuracy of solution as a parameter, with a specified degree of certainty. The class of spherical quadratic programs includes test functions for which other adaptive random search algorithms have reported linear complexity in dimension. Thus the theoretical and computational results describing the near linear search effort of the AMA may provide some intuitive explanation for the performance of similar algorithms. When viewing AMA as an optimizing version of Hit-and-Run, the complexity results give some insight to the value of short sequences while maintaining an iteration count that is linear in dimension.

Intuitively, why might the AMA be linear in dimension? The key to the AMA's performance, from the complexity analysis, is the proportional reduction in volume of the region to be searched for the optimum. This is consistent with the intuition behind the volume reduction of a pure adaptive search, which achieves linear complexity on global optimization problems [23]. Thus the AMA can be expected to exhibit the same agreeable performance on a nonlinear, possibly global, optimization problem with level sets that are approximately spherical in shape. While conventional approaches that rely on the objective function's derivative will most likely be very sensitive to a starting position, and get easily trapped in a

local minimum that may be far from the global minimum. The real usefulness of the AMA is for this type of large dimensional perturbed quadratic program. An example might include a function that is basically convex with minor local minima, such as a function with a sine wave component or an estimation of noisy data, and hence a global optimization problem. This research provides support for the hope that there exists a random search algorithm for global optimization that is linear in dimension.

References

[1] H.C.P. Berbee, C.G.E. Boender, A.H.G. Rinnooy Kan, C.L. Scheffer, R.L. Smith, and J. Telgen, "Hit-and-Run Algorithms for the Identification of Nonredundant Linear Inequalities," *Mathematical Programming 37* (1987) 184-207.

[2] A. Boneh, "A Probabilistic Algorithm for Identifying Redundancy by a Random Feasible Point Generator (RFPG)," in: M.H. Karwan, V. Lotfi, J. Telgen and S. Zionts, eds., *Redundancy in Mathematical Programming* (Springer-Verlag, Berlin, 1983)

[3] G. Cornuejols, M.L. Fisher, and G.L. Nemhauser, "Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms," *Management Science 23* (1977) 789-810.

[4] L.C.W. Dixon and G.P. Szegö, eds., *Towards Global Optimization* (North-Holland, Amsterdam, 1975).

[5] L.C.W. Dixon and G.P. Szegö, eds., *Towards Global Optimization 2* (North-Holland, Amsterdam, 1978).

[6] W. Feller, *An Introduction To Probability Theory And Its Applications, Volume 2, 2nd Edition* (John Wiley and Sons, New York, 1971).

[7] M.H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, eds., *Redundancy in Mathematical Programming* (Springer-Verlag, Berlin, 1983) 108-134.

[8] D.E. Knuth, *The Art of Computer Programming, Vol. 2* (Addison-Wesley, Reading, Massachusetts, 1969) 116.

[9] J.P. Lawrence III and K. Steiglitz, "Randomized pattern search," *IEEE Transactions On Computers C-21* (1972) 382-385.

[10] D.G. Luenberger, *Introduction To Linear And Nonlinear Programming, 2nd Edition* (Addison-Wesley, Reading, Massachusetts, 1984).

[11] K.G. Murty, *Linear Programming* (John Wiley and Sons, New York, 1983).

[12] V.A. Mutseniyeks and L. Rastrigin, "Extremal control of continuous multi-parameter systems by the method of random search," *Engineering Cybernetics 1* (1964) 82-90.

[13] N.R. Patel, R.L. Smith, and Z.B. Zabinsky, "Pure adaptive search in Monte Carlo optimization," *Mathematical Programming 43* (1988) 317-328.

[14] L.A. Rastrigin, "Extremal control by the method of random scanning," *Automation and Remote Control 21* (1960) 891-896.

[15] L.A. Rastrigin, "The convergence of the random method in the extremal control of a many-parameter system," *Automation and Remote Control 24* (1963) 1337-1342.

[16] S.M. Ross, *Stochastic Processes* (John Wiley and Sons, New York, 1983).

[17] G. Schrack and N. Borowski, "An experimental comparison of three random searches," in: F. Lootsma, ed., *Numerical Methods For Nonlinear Optimization* (Academic Press, London, 1972) pp. 137-147.

[18] G. Schrack and M. Choit, "Optimized Relative Step Size Random Searches," *Mathematical Programming 10* (1976) 270-276.

[19] M.A. Schumer and K. Steiglitz, "Adaptive step size random search," *IEEE Transactions On Automatic Control AC-13* (1968) 270-276.

[20] R.L. Smith, "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions," *Operations Research 32* (1984) 1296-1308.

[21] F.J. Solis and R.J.-B. Wets, "Minimization by random search techniques," *Mathematics Of Operations Research 6* (1981) 19-30.

[22] Z.B. Zabinsky, *Computational Complexity Of Adaptive Algorithms In Monte Carlo Optimization* (Dissertation from The University of Michigan, Ann Arbor MI, 1985).

[23] Z.B. Zabinsky and R.L. Smith, "Pure adaptive search in global optimization," *Mathematical Programming* (forthcoming).

[24] W.I. Zangwill, *Nonlinear Programming: A Unified Approach* (Prentice-Hall, New Jersey, 1969).

[25] G. Zoutendijk, *Methods of Feasible Directions* (Elsevier Publishing Company, Amsterdam, 1960).

[26] G. Zoutendijk, "Nonlinear Programming: A Numerical Survey," *SIAM Journal On Control Theory And Applications 4* (1966) 194-210.

Appendix

A Implementation Details

The implementation of the first step of the Adaptive Mixing Algorithm is straightforward. The direction vector D_k is of unit length sampled from a uniform distribution over an n-dimensional hypersphere as follows,

$$D_k = (d_1, d_2, \ldots, d_n) \left(\sum_{i=1}^{n} |x_i|^2 \right)^{-1/2}$$

where $d_i, i = 1, 2, \ldots, n$ are sampled independently from a standard normal distribution, $N(0,1)$ [8].

The implementation of Step 2 is more involved. The implementation used in our computational experiments assumed the original problem had linear constraints, as follows:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax \leq b \end{array}$$

where $f(x)$ is a continuous function, A is an $m \times n$ matrix, x is an n-dimensional vector, and b is an m-dimensional vector. Step 2 generates X_{k+1} uniformly over L_k using a line search. Line search methods are themselves an important area of research [10], but will be kept simple here. It proceeds as follows.

First, the direction D_k is checked to be improving and feasible a distance ϵ away by evaluating the following equations:

$$\begin{array}{lll} (1) & f(X_k + \epsilon D_k) < f(X_k), & \text{and} \\ (2) & X_k + \epsilon D_k \in S, & \end{array}$$

where ϵ represents a small step (the value of ϵ is a parameter in the computer program, $\epsilon = 10^{-5}$ was used in most experiments). If both equations are satisfied, then a step of length ϵ in direction D_k is possible. Equation (1) determines if direction D_k is improving. If it is not satisfied, then $-D_k$ is checked. In the general case, for regions that are differentiable at X_k, the probability that both directions $\pm D_k$ are not improving approaches zero as ϵ approaches zero. There is a positive probability that both directions are not improving if the region is not differentiable at X_k, or ϵ is large relative to the curvature of the improving region at X_k. If no improving direction is found after a maximum number of tries, the computer program terminates with a message to the user. Equation (2) determines if an ϵ step in direction D_k is feasible. In particular, if it is not satisfied, the algorithm returns to Step 1 to generate another direction. Because the points are chosen randomly, the probability of landing on

a constraint is zero; however, the point may be within ϵ of the edge in direction D_k. This problem of "jamming" seemed especially evident in solving linear programs. Zangwill [24] (see also [25,26]) discusses jamming, where the points cluster in a corner of the feasible region, in the context of feasible direction methods. These authors also discuss antijamming procedures such as a perturbation method. Such an antijamming technique has not been incorporated into this version of the Adaptive Mixing Algorithm, although it is intended for future research.

Given an improving feasible direction D_k, the line search next generates an improving feasible point X_{k+1} uniformly within L_k satisfying:

$$\begin{array}{ll} (3) & f(X_{k+1}) < f(X_k), \qquad \text{and} \\ (4) & X_{k+1} \in S. \end{array}$$

To ensure feasibility (equation (4)) for the problem with linear constraints, a minimum ratio test is used to determine the maximum feasible distance, λ, in direction D_k such that $X_k + \theta D_k$ is feasible for all $0 \leq \theta \leq \lambda$ [11]. Then, a random distance δ is generated according to a uniform distribution on $(0, \lambda)$ until equation (3) is satisfied. Thus the new point $X_k + \delta D_k$ is improving, and is accepted as X_{k+1}. One way to increase the efficiency of the line search for problems with convex objective functions is to retain the value of δ as $\lambda = \delta$, and then generate another distance until an improving point is found. This modified line search will still produce uniformly distributed improving points due to the nesting of the convex level sets. This modification was used in the two computational experiments reported earlier. There are many possible line searches that may be more efficient than this simple line search, but our analysis in terms of number of iterations is valid for any line search that implements Step 2 of the algorithm.

B Lemma 3.0

Lemma 3.0 *Given the Adaptive Mixing Algorithm and any conical spherical quadratic program (Q), the ratios of normalized improvements are independent and identically distributed,*

$$Z_{k+1}/Z_k \sim \text{i. i. d.} \qquad \text{for } k = 0, 1, 2, \ldots$$

Let the mean and variance of the ratios *of normalized improvement be written as,*

$$E[Z_{k+1}/Z_k] = \mu, \qquad \text{and} \qquad Var[Z_{k+1}/Z_k] = \sigma^2,$$

then the mean and variance of the normalized improvement is,

$$E[Z_k] \;=\; \mu^k$$

$$
\begin{aligned}
Var[Z_k] &= (\sigma^2+\mu^2)^k - \mu^{2k} \\
&= \left(\sigma^2+\mu^2\right)^k \left(1 - \left(\frac{\mu^2}{\sigma^2+\mu^2}\right)^k\right).
\end{aligned}
$$

Proof: First we reiterate that this proof applies to any conical spherical quadratic program (Q). Then, for $k = 0, 1, 2, \ldots$ and $0 \leq z \leq 1$

$$
\begin{aligned}
P(Z_{k+1}/Z_k \leq z) &= E_{Z_k}[P(Z_{k+1} \leq zZ_k | Z_k)] \\
&= \int_0^1 P(Z_{k+1} \leq zx | Z_k = x) f_{Z_k}(x) dx,
\end{aligned}
$$

where $f_{Z_k}(x)$ is the density function of Z_k, and since $Z_k = (Y_k - y_*)/(y^* - y_*)$

$$
= \int_0^1 P(Y_{k+1} \leq (zx)' | Y_k = (x)') f_{Z_k}(x) dx,
$$

where $(x)' = x(y^* - y_*) + y_*$ and $(zx)' = zx(y^* - y_*) + y_*$.

As discussed in Section 3, the conditional probability of improvement on (Q) is invariant and depends on the ratio of the lengths of line segments,

$$
P(Y_{k+1} \leq (zx)' | Y_k = (x)') = \int_d \|L_{Q_{(zx)'}}(d,p)\| / \|L_{Q_{(x)'}}(d,p)\| dF_D(d)
$$

for *any* point p with $f(p) = (x)'$. Now, since the ratio of lengths is preserved by a linear tranformation, this ratio is constant for all $0 < x < 1$, and depends only on the scaling factor z. Thus the conditional probability $P(Y_{k+1} \leq (zx)' | Y_k = (x)')$ is constant for all x and depends only on z. We write

$$
P(Y_{k+1} \leq (zx)' | Y_k = (x)') = L(z)
$$

Using this notation in the expression above,

$$
\begin{aligned}
P(Z_{k+1}/Z_k \leq z) &= \int_0^1 P(Y_{k+1} \leq (zx)' | Y_k = (x)') f_{Z_k}(x) dx \\
&= \int_0^1 L(z) f_{Z_k}(x) dx \\
&= L(z) \int_0^1 f_{Z_k}(x) dx \\
&= L(z)
\end{aligned}
$$

for $0 \leq z \leq 1$. Thus the random variables Z_{k+1}/Z_k for $k = 0, 1, 2, \ldots$ are identically distributed.

To establish independence of Z_{k+1}/Z_k, we need to show that

$$P\left(Z_{k+1}/Z_k \leq z | Z_k/Z_{k-1}, \ldots, Z_2/Z_1, Z_1/Z_0\right) = P\left(Z_{k+1}/Z_k \leq z\right)$$

for all $k = 0, 1, 2, \ldots$ and $0 \leq z \leq 1$. Now, since $Z_0 = 1$ we have, for $0 \leq z \leq 1$ and $0 \leq z_i \leq 1$, $i = 1, 2, \ldots, k$ and for all $k = 0, 1, 2, \ldots$,

$$P\left(Z_{k+1}/Z_k \leq z | Z_k/Z_{k-1} = z_k, \ldots, Z_1/Z_0 = z_1\right)$$

$$\begin{aligned} &= P\left(Z_{k+1} \leq z Z_k | Z_k = z_k z_{k-1} \cdots z_1, \ldots, Z_1 = z_1\right) \\ &= P\left(Z_{k+1} \leq z z_k z_{k-1} \cdots z_1 | Z_k = z_k z_{k-1} \cdots z_1\right) \end{aligned}$$

since Z_{k+1} is conditionally independent of $Z_{k-1}, \ldots, Z_1$ given Z_k

$$= P\left(Z_{k+1} \leq zx | Z_k = x\right)$$

where $x = z_k z_{k-1} \cdots z_1$

$$\begin{aligned} &= P\left(Y_{k+1} \leq (zx)' | Y_k = (x)'\right) \\ &= L(z) \\ &= P\left(Z_{k+1}/Z_k \leq z\right). \end{aligned}$$

Thus the random variables Z_{k+1}/Z_k for $k = 0, 1, 2, \ldots$ are independent.

Now that Z_{k+1}/Z_k have been shown to be independent and identically distributed, let μ and σ^2 be the mean and variance, respectively for the distribution. Notice the mean and variance do not depend on the value of k. We now derive the mean and variance for the random variable Z_k, as a function of k, μ, and σ^2. Since $Z_0 = 1$,

$$Z_k = \left(Z_k/Z_{k-1}\right)\left(Z_{k-1}/Z_{k-2}\right) \cdots \left(Z_1/Z_0\right)$$

and by independence,

$$\begin{aligned} E\left[Z_k\right] &= E\left[Z_k/Z_{k-1}\right] E\left[Z_{k-1}/Z_{k-2}\right] \cdots E\left[Z_1/Z_0\right] \\ &= \mu^k. \end{aligned}$$

Similarly, the second moment can be written,

$$E\left[Z_k^2\right] = E\left[\left(Z_k/Z_{k-1}\right)^2\right] E\left[\left(Z_{k-1}/Z_{k-2}\right)^2\right] \cdots E\left[\left(Z_1/Z_0\right)^2\right].$$

Also, the

$$Var\left[Z_{k+1}/Z_k\right] = \sigma^2 = E\left[\left(Z_{k+1}/Z_k\right)^2\right] - \mu^2$$

implies that, for $k = 0, 1, 2, \ldots$

$$E\left[(Z_{k+1}/Z_k)^2\right] = \sigma^2 + \mu^2,$$

hence

$$E\left[Z_k^2\right] = \left(\sigma^2 + \mu^2\right)^k.$$

Thus,

$$\begin{aligned} Var\left[Z_k\right] &= E\left[Z_k^2\right] - E\left[Z_k\right]^2 \\ &= \left(\sigma^2 + \mu^2\right)^k - \mu^{2k} \end{aligned}$$

and factoring out the $(\sigma^2 + \mu^2)^k$ term, we conclude with

$$= \left(\sigma^2 + \mu^2\right)^k \left(1 - \left(\frac{\mu^2}{\sigma^2 + \mu^2}\right)^k\right). \qquad \blacksquare$$

C Theorem 3.1

Theorem 3.1 *Given the Adaptive Mixing Algorithm and any spherical quadratic program* (P) *with* $\mu(n)$ *for its conical program* (Q)*, then for any* $m > 1$ *and* $0 < \alpha < 1$*,*

$$K_{\alpha,m} \leq \frac{2\ln\left(m(1 + \alpha^{-1/2})\right)}{\ln(1/\mu(n))}.$$

Proof: The proof is separated into four parts that establish that (Q) is a more difficult problem than (P), and uses (Q) to provide an upper bound on $K_{\alpha,m}$. The superscripts P and Q are used in the notation to specify the problem. The first part shows conditional dominance,

$$P\left(Z_{k+1}^Q \leq z | Z_k^Q = w\right) \leq P\left(Z_{k+1}^P \leq z | Z_k^P = w\right)$$

for all $k = 0, 1, 2, \ldots$ and $0 \leq z, w \leq 1$. This is used in part 2 to show stochastic dominance,

$$\left\{Z_{k+1}^Q\right\} \geq_{st} \left\{Z_{k+1}^P\right\}$$

for all $k = 1, 2, \ldots$ which is used in part 3 to show that (Q) is stochastically more complex than (P),

$$K_{\alpha,m}^Q \geq K_{\alpha,m}^P$$

for any $m > 1$ and $0 < \alpha < 1$. Part 4 establishes an upper bound on the iteration count for (Q), and combining with part 3 yields

$$K^P_{\alpha,m} \leq K^Q_{\alpha,m} \leq \frac{2\ln\left(m(1+\alpha^{-1/2})\right)}{\ln(1/\mu(n))}$$

which completes the proof of the theorem.

Part 1 of proof: Show conditional dominance, i.e., show that

$$P\left(Z^Q_{k+1} \leq z | Z^Q_k = w\right) \leq P\left(Z^P_{k+1} \leq z | Z^P_k = w\right)$$

for all $k = 0, 1, 2, \ldots$ and $0 \leq z, w \leq 1$. First consider the case when $0 \leq w \leq z \leq 1$. Then, by the definition of the Adaptive Mixing Algorithm,

$$P\left(Z^Q_{k+1} \leq z | Z^Q_k = w\right) = 1 = P\left(Z^P_{k+1} \leq z | Z^P_k = w\right)$$

for $k = 0, 1, 2, \ldots$. Second consider the more interesting case when $0 \leq z < w \leq 1$. Now, using the definition of Z_k, clearly for either problem (P) or (Q), and for $k = 0, 1, 2, \ldots$

$$P\left(Z_{k+1} \leq z | Z_k = w\right) = P\left(Y_{k+1} \leq z' | Y^P_k = w'\right)$$

where $z' = z(y^* - y_*) + y_*$ and $w' = w(y^* - y_*) + y_*$. It then remains to show that

$$P\left(Y^Q_{k+1} \leq z | Y^Q_k = w\right) = 1 = P\left(Y^P_{k+1} \leq z | Y^P_k = w\right)$$

for $k = 0, 1, 2, \ldots$ and $y_* \leq z < w \leq y^*$.

In order to prove this equivalent statement, we need to introduce a similarity transformation and additional notation. Let $\lambda_{w,z} : \mathbf{R}^n \to \mathbf{R}^n$ be an affine function defined by

$$\lambda_{w,z}(x) = x_* + \left((z - y_*)/(w - y_*)\right)(x - x_*)$$

for $y_* \leq z < w \leq y^*$.

Let $\tilde{P}_{w,z}$ be the level set P_w shrunken by a factor of $(z - y_*)/(w - y_*)$ and rerooted at x_* so that all $\tilde{P}_{w,z}$ is contained in P_z, i.e.,

$$\tilde{P}_{w,z} = \{\tilde{x} : \tilde{x} = \lambda_{w,z}(x) \text{ for } x \in P_w\}.$$

Similarly, let

$$\tilde{Q}_{w,z} = \{\tilde{x} : \tilde{x} = \lambda_{w,z}(x) \text{ for } x \in Q_w\}.$$

Notice that $\tilde{P}_{w,z}$ is contained in P_z, and $\tilde{Q}_{w,z} = Q_z$ (the proofs are in [13]). Using this, we have

$$P\left(Y_{k+1}^Q \leq z | Y_k^Q = w\right) = P\left(X_{k+1}^Q \in \tilde{Q}_{w,z} | Y_k^Q = w\right) \tag{1}$$

and

$$P\left(Y_{k+1}^P \leq z | Y_k^P = w\right) \geq P\left(X_{k+1}^P \in \tilde{P}_{w,z} | Y_k^P = w\right). \tag{2}$$

Now,

$$P\left(X_{k+1}^P \in \tilde{P}_{w,z} | Y_k^P = w\right) = \int_d \|L_{\tilde{P}_{w,z}}(d,p)\| / \|L_{P_w}(d,p)\| dF_D(d),$$

for any point p with $g(p) = w$, and similarly,

$$P\left(X_{k+1}^Q \in \tilde{Q}_{w,z} | Y_k^Q = w\right) = \int_d \|L_{\tilde{Q}_{w,z}}(d,q)\| / \|L_{Q_w}(d,q)\| dF_D(d),$$

for any point q with $g(q) = w$. Due to the linear similarity transformation $\lambda_{w,z}$, for every d that intersects $\tilde{P}_{w,z}$, the corresponding d also intersects $\tilde{Q}_{w,z}$. Also, for any feasible improving direction d, the linear similarity transformation preserves the ratio of the lengths of these corresponding line segments,

$$\|L_{\tilde{P}_{w,z}}(d,p)\| / \|L_{P_w}(d,p)\| = \|L_{\tilde{Q}_{w,z}}(d,q)\| / \|L_{Q_w}(d,q)\|.$$

Therefore,

$$P\left(X_{k+1}^Q \in \tilde{Q}_{w,z} | Y_k^Q = w\right) = P\left(X_{k+1}^P \in \tilde{P}_{w,z} | Y_k^P = w\right),$$

and using equations (1) and (2) above, we have the desired result

$$P\left(Y_{k+1}^Q \leq z | Y_k^Q = w\right) \leq P\left(Y_{k+1}^P \leq z | Y_k^P = w\right)$$

for $k = 0, 1, 2, \ldots$ and $y_* \leq z < w \leq y^*$.

Part 2 of proof: Show stochastic dominance, i.e., show that

$$\left\{Z_{k+1}^Q\right\} \geq_{st} \left\{Z_{k+1}^P\right\}$$

for all $k = 1, 2, \ldots$, or equivalently,

$$P\left(Z_k^Q > z\right) \geq P\left(Z_k^P > z\right)$$

for all $k = 1, 2, \ldots$ and $0 \leq z \leq 1$.

The proof uses induction on k. To show the first iteration, recall that the initial values are defined to be equal to 1, $Z_0^Q = Z_0^P = 1$. Thus for any z, $0 \leq z \leq 1$,

$$\begin{aligned} P\left(Z_1^Q > z\right) &= P\left(Z_1^Q > z | Z_0^Q = 1\right) \\ &\geq P\left(Z_1^P > z | Z_0^P = 1\right) \end{aligned}$$

by conditional dominance (part 1)

$$= P\left(Z_1^P > z\right).$$

Therefore, $\left\{Z_1^Q\right\} \geq_{st} \left\{Z_1^P\right\}$.

To continue the induction argument, we assume that $Z_k^Q \geq_{st} Z_k^P$ for some integer $k > 1$, and show that $Z_{k+1}^Q \geq_{st} Z_{k+1}^P$. For any fixed z, $0 \leq z \leq 1$, let

$$\begin{aligned} q(w) &= P\left(Z_{k+1}^Q > z | Z_k^Q = w\right), \qquad \text{and} \\ p(w) &= P\left(Z_{k+1}^P > z | Z_k^P = w\right) \end{aligned}$$

for $0 \leq w \leq 1$. Then, $q(w) \geq p(w)$ for $0 \leq w \leq 1$ by conditional dominance (part 1). Also, $q(w)$ is a non-decreasing function because the conditional distribution of improvement is non-increasing in w. (This is a technical point but can be readily proved using properties of a cumulative distribution function and the fact that Z_{k+1}^Q / Z_k^Q are independent and identically distributed as shown in Lemma 1.) Thus, because $q(w)$ is non-decreasing and $Z_k^Q \geq_{st} Z_k^P$, we have [16]

$$\begin{aligned} E\left[q(Z_k^Q)\right] &\geq E\left[q(Z_k^P)\right] \\ &\geq E\left[p(Z_k^P)\right] \end{aligned}$$

by $q(w) \geq p(w)$ for $0 \leq w \leq 1$, and thus

$$E\left[q(Z_k^Q)\right] \geq E\left[p(Z_k^P)\right].$$

Now,

$$E\left[q(Z_k^Q)\right] = E\left[P\left(Z_{k+1}^Q > z | Z_k^Q\right)\right] = P\left(Z_{k+1}^Q > z\right)$$

and similarly,

$$E\left[p(Z_k^P)\right] = E\left[P\left(Z_{k+1}^P > z | Z_k^P\right)\right] = P\left(Z_{k+1}^P > z\right).$$

Thus, $P\left(Z_{k+1}^Q > z\right) \geq P\left(Z_{k+1}^P > z\right)$ for all $0 \leq z \leq 1$, and therefore $\left\{Z_{k+1}^Q\right\} \geq_{st} \left\{Z_{k+1}^P\right\}$.

By induction, we have $\left\{Z_k^Q\right\} \geq_{st} \left\{Z_k^P\right\}$ for all $k = 1, 2, \ldots$.

Part 3 of proof: Show that (Q) is stochastically more complex than (P), i.e., show that

$$K^Q_{\alpha,m} \geq K^P_{\alpha,m}$$

for any $m > 1$ and $0 < \alpha < 1$. By stochastic dominance (part 2 of the proof), we have that

$$P\left(Z^Q_k \leq z\right) \leq P\left(Z^P_k \leq z\right)$$

for $k = 1, 2, \ldots$ and $0 \leq z \leq 1$, and letting $z = 1/m$ implies

$$P\left(Z^Q_k \leq 1/m\right) \leq P\left(Z^P_k \leq 1/m\right)$$

for $k = 1, 2, \ldots$ and $m > 1$. Let $M = K^Q_{\alpha,m} = \min_{k=1,2,\ldots}\left\{k : P\left(Z^Q_k \leq 1/m\right) \geq 1 - \alpha\right\}$ for $m > 1$ and $0 < \alpha < 1$. Then

$$P\left(Z^P_M \leq 1/m\right) \geq P\left(Z^Q_M \leq 1/m\right) \geq 1 - \alpha$$

hence,

$$K^P_{\alpha,m} = \min_{k=1,2,\ldots}\left\{k : P\left(Z^P_k \leq 1/m\right) \geq 1 - \alpha\right\} \leq M.$$

Therefore, $K^Q_{\alpha,m} \geq K^P_{\alpha,m}$ for any $m > 1$ and $0 < \alpha < 1$.

Part 4 of proof: For any $m > 1$ and $0 < \alpha < 1$, establish an upper bound on the complexity measure on (Q) as follows,

$$K^Q_{\alpha,m} \leq \frac{2\ln\left(m(1+\alpha^{-1/2})\right)}{\ln(1/\mu(n))}.$$

To establish this upper bound, let $\mu_k = E\left[Z^Q_k\right]$, and $\sigma^2_k = Var\left[Z^Q_k\right]$. From Chebyshev's inequality [6],

$$P\left(|Z^Q_k - \mu_k| < t\right) \geq 1 - \sigma^2_k/t^2$$

for any $t > 0$. Letting $t^2 = \sigma^2_k/\alpha$ for $0 < \alpha < 1$, implies

$$\begin{aligned} P\left(|Z^Q_k - \mu_k| < \sigma_k\alpha^{-1/2}\right) &\geq 1 - \alpha, \qquad \text{or} \\ P\left(\mu_k - \sigma_k\alpha^{-1/2} < Z^Q_k < \mu_k + \sigma_k\alpha^{-1/2}\right) &\geq 1 - \alpha. \end{aligned}$$

Now, using the definition of complexity,

$$K^Q_{\alpha,m} = \min_{k=1,2,\ldots}\left\{k : P\left(Z^Q_k \leq 1/m\right) \geq 1 - \alpha\right\}$$

to bound complexity we seek k such that $P\left(Z_k^Q \leq 1/m\right) \geq 1-\alpha$. Since $0 \leq z \leq 1$,

$$P\left(Z_k^Q \leq 1/m\right) \geq P\left(0 \leq Z_k^Q \leq 1/m\right)$$

and letting $1/m \geq \mu_k + \sigma_k \alpha^{-1/2}$,

$$\begin{aligned} &\geq P\left(0 \leq Z_k^Q \leq \mu_k + \sigma_k \alpha^{-1/2}\right) \\ &\geq P\left(\mu_k - \sigma_k \alpha^{-1/2} \leq Z_k^Q \leq \mu_k + \sigma_k \alpha^{-1/2}\right). \end{aligned}$$

Combining this with the expression from Chebyshev's inequality yields, if k is such that $1/m \geq \mu_k + \sigma_k \alpha^{-1/2}$, then $P\left(Z_k^Q \leq 1/m\right) \geq 1-\alpha$, and hence $K_{\alpha,m}^Q \leq k$.

Now, using Lemma 3.0, and the expressions given for μ_k and σ_k^2,

$$\begin{aligned} \mu_k &= E[Z_k] &&= \mu^k \\ \sigma_k^2 &= Var[Z_k] &&= (\sigma^2+\mu^2)^k - \mu^{2k} \\ & &&= (\sigma^2+\mu^2)^k \left(1 - \left(\tfrac{\mu^2}{\sigma^2+\mu^2}\right)^k\right). \end{aligned}$$

we next perform a series of substitutions to find some k that satisfies

$$1/m \geq \mu_k + \sigma_k \alpha^{-1/2}$$

or equivalently, we seek k that satisfies

$$1/m \geq \mu^k + \sqrt{(\sigma^2+\mu^2)^k \left(1 - \left(\frac{\mu^2}{\sigma^2+\mu^2}\right)^k\right)/\alpha}.$$

Because $0 \leq \mu \leq 1$ and $\sigma^2 > 0$, we have

$$\begin{aligned} \sigma^2 + \mu^2 > \mu^2 &\geq 0, && \text{which implies} \\ \mu^2/\left(\sigma^2+\mu^2\right)^k &\geq 0, && \text{which implies} \\ 1 - \left(\mu^2/\left(\sigma^2+\mu^2\right)\right)^k &\leq 1. \end{aligned}$$

Therefore, if k' satisfies

$$\begin{aligned} 1/m &\geq \mu^{k'} + \sqrt{(\sigma^2+\mu^2)^{k'}/\alpha}, && \text{then} \\ 1/m &\geq \mu^{k'} + \sqrt{(\sigma^2+\mu^2)^{k'} \left(1 - \left(\frac{\mu^2}{\sigma^2+\mu^2}\right)^{k'}\right)/\alpha}, \end{aligned}$$

then $k' \geq K^Q_{\alpha,m}$.
Because $0 \leq \mu \leq 1$, we have for $k = 0, 1, 2, \ldots$

$$\mu^k \leq \mu^{k/2}.$$

Therefore, if k'' satisfies

$$1/m \;\geq\; \mu^{k''/2} + \sqrt{\left(\sigma^2 + \mu^2\right)^{k''}/\alpha},$$

then $k'' \geq K^Q_{\alpha,m}$.
Now, $\sigma^2 + \mu^2 < \mu$,which implies for $k = 0, 1, 2, \ldots$, we have

$$\left(\sigma^2 + \mu^2\right)^{k/2} \geq \left(\mu^2\right)^{k/2}.$$

Therefore, if k''' satisfies

$$\begin{aligned} 1/m &\geq \mu^{k'''/2} + \mu^{k'''/2}\sqrt{1/\alpha}, \qquad \text{or} \\ 1/m &\geq \mu^{k'''/2}\left(1 + \alpha^{-1/2}\right) \end{aligned}$$

Letting $\kappa = k'''$ (for convenience in notation), and manipulating the last equation gives,

$$1 \geq m\mu^{\kappa/2}\left(1 + \alpha^{-1/2}\right),$$

which implies

$$0 = \ln(1) \geq (\kappa/2)\ln(\mu) + \ln\left(m\left(1 + \alpha^{-1/2}\right)\right),$$

which implies

$$-(\kappa/2)\ln(\mu) \geq \ln\left(m\left(1 + \alpha^{-1/2}\right)\right)$$

which implies

$$\kappa \leq \frac{-2\ln\left(m\left(1 + \alpha^{-1/2}\right)\right)}{\ln(\mu)} = \frac{2\ln\left(m\left(1 + \alpha^{-1/2}\right)\right)}{\ln(1/\mu)}.$$

Therefore, if $\kappa = 2\ln\left(m\left(1 + \alpha^{-1/2}\right)\right)/\ln(1/\mu)$, then

$$1/m \;\geq\; \mu_\kappa + \sigma_\kappa \alpha^{-1/2},$$

thus $P\left(Z^Q_k \leq 1/m\right) \geq 1 - \alpha$. Hence, κ provides an upper bound for $K^Q_{\alpha,m}$ and, combining this bound with the bound on $K^P_{\alpha,m}$ from part 3, we have

$$K^P_{\alpha,m} \leq K^Q_{\alpha,m} \leq \frac{2\ln\left(m\left(1 + \alpha^{-1/2}\right)\right)}{\ln(1/\mu)}$$

for any $m > 1$ and $0 < \alpha < 1$, which completes the proof. ∎

n	$\hat{\mu}(n)$
2	0.910
4	0.945
6	0.956
8	0.965
10	0.969
20	0.986
30	0.991
40	0.994
50	0.994

Table 1: Estimates For Expected Ratio Of Normalized Improvements ($\hat{\mu}(n)$) As A Function Of Dimension (n)

n	$K_{0.01,10^6}$
1	218
2	331
5	672
10	1,239
50	5,779
100	11,454
500	56,851
1,000	113,598
5,000	567,573
10,000	1,135,043

Table 2: Bound On Complexity For The Adaptive Mixing Algorithm ($K_{0.01,10^6}$) As A Function Of The Number Of Dimensions (n)

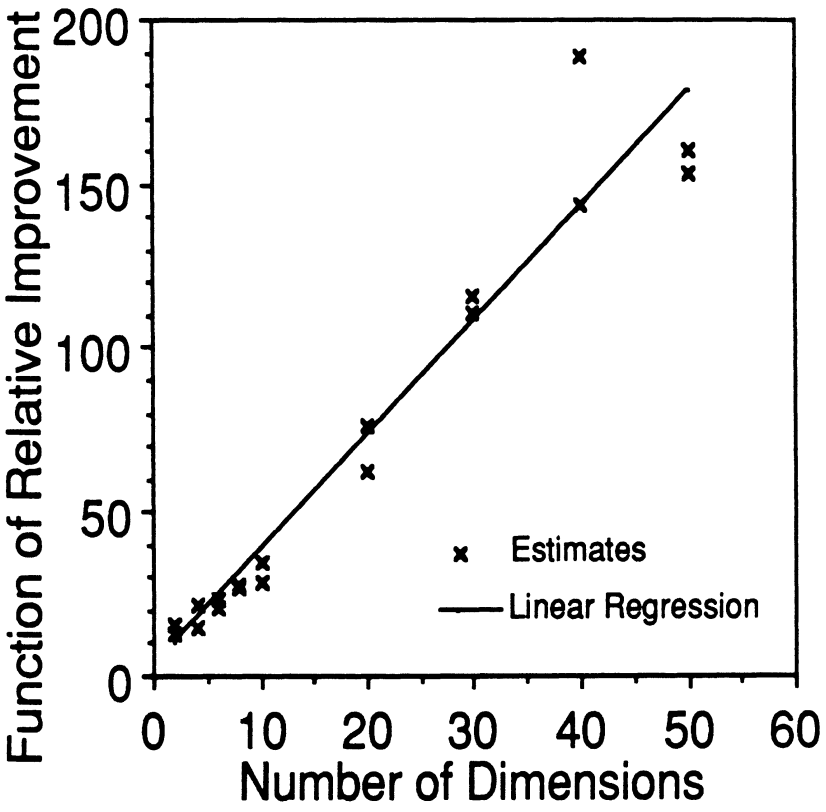

Figure 1: Estimates Of Average Normalized Improvement Versus Dimension

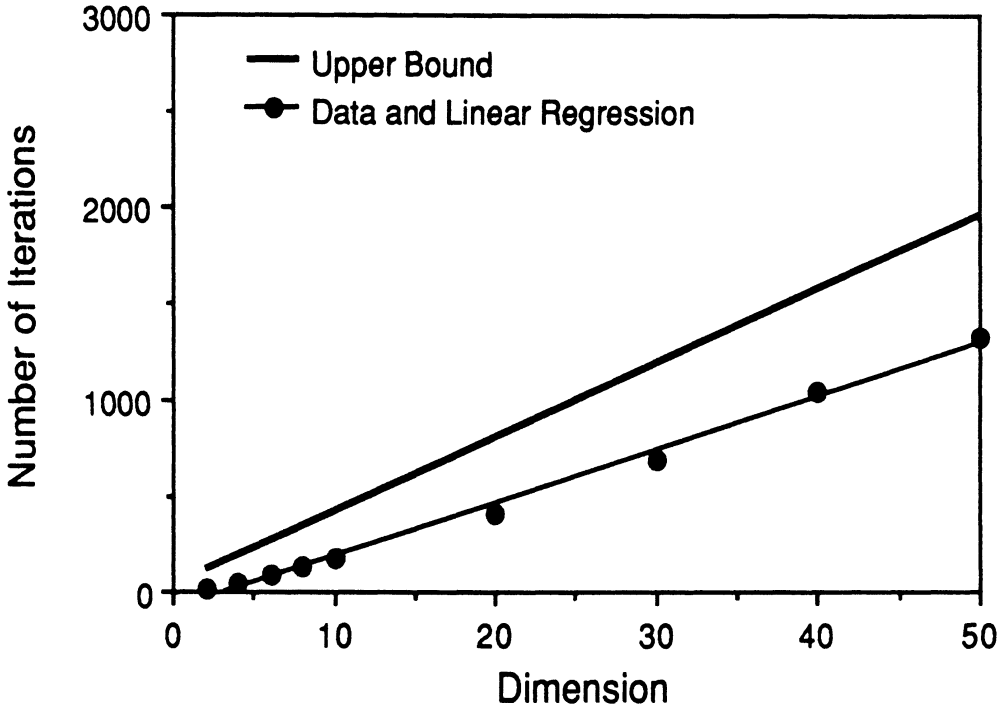

Figure 2: Average Number of Iterations for AMA On The Conical Program ($Q.1$).

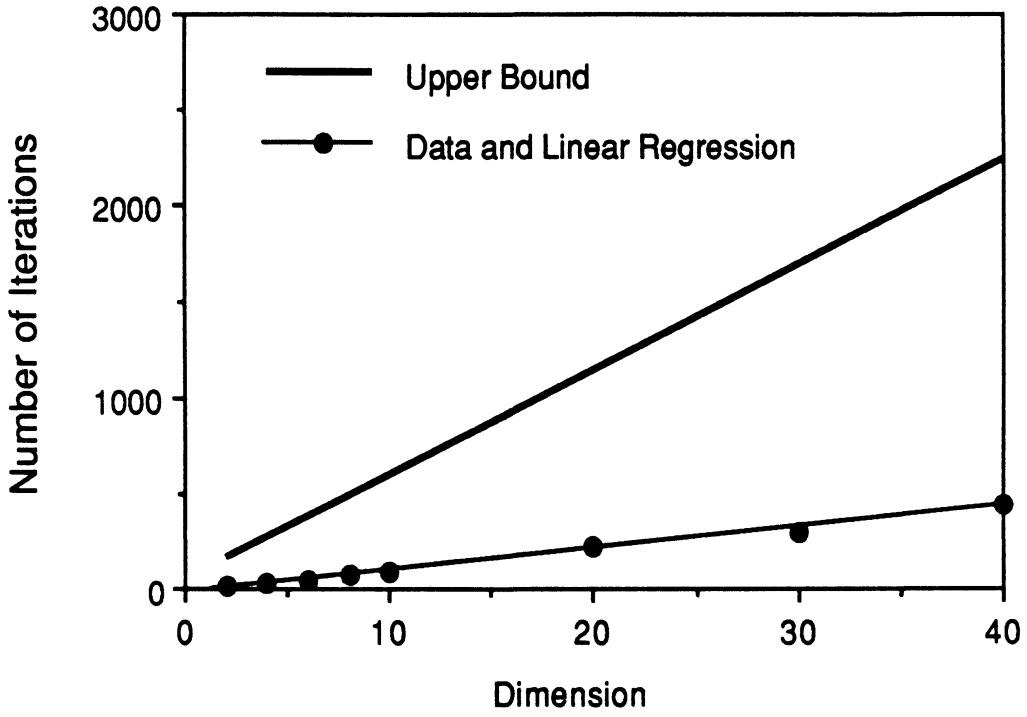

Figure 3: Average Number of Iterations for AMA On The Hyperspherical Problem ($P.1$).