Evolutionary Pattern Search Algorithms*

William E. Hart
Applied and Numerical Mathematics Department
Sandia National Labs
P. O. Box 5800 - MS 1110
Albuquerque, NM 87185-1110
wehart@cs.sandia.gov

September 19, 1995

Abstract

This paper defines a class of evolutionary algorithms called evolutionary pattern search algorithms (EPSAs) and analyzes their convergence properties. This class of algorithms is closely related to evolutionary programming, evolutionary strategie and real-coded genetic algorithms. EPSAs are self-adapting systems that modify the step size of the mutation operator in response to the success of previous optimization steps. The rule used to adapt the step size can be used to provide a stationary point convergence theory for EPSAs on any continuous function. This convergence theory is based on an extension of the convergence theory for generalized pattern search methods. An experimental analysis of the performance of EPSAs demonstrates that these algorithms can perform a level of global search that is comparable to that of canonical EAs. We also describe a stopping rule for EPSAs, which reliably terminated near stationary points in our experiments. This is the first stopping rule for any class of EAs that can terminate at a given distance from stationary points.

*This work was supported by the Applied Mathematical Sciences program, U.S. Department of Energy, Office of Energy Research, and was performed at Sandia National Laboratories, operated for the U.S. Department of Energy under contract No. DE-AC04-94AL85000.

1 Introduction

This paper concerns the application of evolutionary search algorithms to solve an unconstrained minimization problem to find $x^* \in D$ such that

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

where $D \subset \mathrm{R}^n$ and $f : D \rightarrow \mathrm{R}$. In particular, this paper concerns the convergence properties of a class of evolutionary algorithms (EAs) that is closely related to evolutionary programing (EP), evolutionary strategie (ES) and real-coded genetic algorithms (GAs), which use real-coded (floating-point) genes as opposed to binary-coded genes. The class of evolutionary algorithms that I analyze encompasses EP, ES and GAs to the extent that both mutation and crossover may be stochastically applied to generate new solutions. A distinguishing feature of these EAs is their use of a mutation operator that generates a discrete number of possible values at each allele. This discretization makes it is possible to adapt the step size to guarantee convergence to a stationary point of the objective function, where the derivative is zero, with probability one.

To prove this convergence result, I define a convergence theory for generalized stochastic pattern search methods, which are an extension of the generalized pattern search methods described by Torczon [31]. Like EAs, pattern search methods are optimization algorithms that examine exploratory moves in search of solutions with lower functional values. In stochastic pattern search methods the exploratory moves may be stochastically selected. The class of EAs that I analyze, which I call evolutionary pattern search algorithms (EPSAs), can be formulated as a stochastic pattern search method, so the convergence analysis for stochastic pattern search methods applies to these algorithms.

The convergence analysis for EPSAs has a number of important implications. First, EPSAs are guaranteed to converge on any continuous function. Thus EPSAs can optimize nonsmooth functions, which is an advantage of EAs in general over gradient-based optimization methods. Second, the fact that EPSAs converge to stationary points allows us to define stopping rules that terminate when the EPSA is near the stationary point. To my knowledge, the stopping rule described for EPSAs is the first stopping rule for any EA that can successfully terminate near a stationary point. Finally, the effects of the relative frequency of mutation and crossover can be used to provide a guideline for determining the length of time EPSAs will need to terminate.

While EAs are typically described as methods for global optimization, this convergence analysis does not guarantee that the global optimum is found. However, pattern search algorithms have been successfully applied to global optimization problems. For example, Meza and Martinez [17] have successfully applied a pattern search method, parallel directed search, to the global optimization of the conformational energy of a simple chain molecule. Their comparison of parallel directed search to GA and simulated annealing indicates that parallel directed search is equally effective at performing global optimization for this problem. I present experimental results for EPSAs that confirms that their ability to perform global optimization comparable to that of canonical EAs.

This paper is organized as follows. Section 2 provides some background on EP, ES and GAs, and reviews convergence theories for these EAs. Section 3 defines generalized pattern search methods and reviews the convergence theorems from Torczon [31]. Section 4 defines generalized stochastic pattern search methods and proves generalizations of these convergence theorems for this class of algorithms. Section 5 presents a class of EAs that

fits within the framework of stochastic pattern search algorithms and analyzes the relative complexity of EPSAs using different genetic operators. Section 6 presents some numerical results that evaluate the performance of these EAs on global optimization problems. Finally, I describe extensions of the basic formulation of EPSAs and discuss some implications of these results.

Notation I will denote by R,Q and Z the sets of real, rational and integer numbers, respectively. All norms will be Euclidean vector norms or the associated operator norm. I also define $L(x_0) = \{x \mid f(x) \leq f(x_0)\}$.

2 Background

2.1 A Canonical Evolutionary Algorithm

We consider EAs that optimize a *fitness function* of the form $f : D \rightarrow \mathbf{R}^n$, $D \subset \mathbf{R}$. We assume that the function f is to be minimized. The *population* used by an EA consists of a N-tuple of vectors $x_i \in \mathbf{R}^n$. Each vector x_i, called an *individual* of the population, is a feasible solution to the problem, and the value $f(x_i)$ is said to be the *fitness* of the solution. Let $\{x_1^t, \ldots, x_N^t\}$ be the N individuals in a population at time t.

The following pseudo-code describes the steps executed in a canonical EA that encompasses EP, ES and real-coded GAs:

1 Select an initial population $\{x_1^0, \ldots, x_N^0\}$, $x_i^0 \in \mathbf{R}^n$
2 Determine the fitness of each individual
3 Repeat $t = 1, 2, \ldots$
4 Perform selection
5 Perform crossover with probability χ
6 Perform mutation with probability μ
7 Determine the fitness of each individual
8 $x_t^* = \arg\min_{i=1,\ldots,N} f(x_i^t)$ and $y_t^* = f(x_t^*)$
9 Perform replacement
10 Until some stopping criterion is satisfied

There are a number of details of EP, ES and real-coded GAs that distinguish them from these EAs, but this algorithmic framework captures the principle features of these algorithms. For further details of these algorithms see Fogel [8, 9] for descriptionss of EP, Bäck and Schwefel [3] and Bäck, Hoffmeister and Schwefel [1] for descriptions of ES, and Goldberg [10] and Davis [4] for descriptions of GAs. The principle feature distinguishing these EAs is the selection of operators used perform the evolutionary search. EP utilizes mutation to generate new solutions. ES and GAs utilize both mutation and crossover, although crossover is applied with relatively greater frequency in GAs. For our purposes, I call a canonical EA an EP if $\chi = 0$, an ES if $\chi > 0$ and $\mu >> 0$. and a real-coded GA if $\chi \gg 0$ and $\mu \approx 0$.

With high probability, the selection algorithm chooses the solutions from the current population that minimize the error, $\|f(x) - f(x^*)\|$. Typically, the solutions with the highest error are chosen with a low, non-zero probability. Consequently, an EA's search is concentrated in regions with near-optimal fitness, though regions with poor fitness are still searched with low probability. Goldberg and Deb [12] review a variety of selection algorithms commonly used in GAs.

Crossover and mutation are operators used to construct new solutions from the individuals chosen by the selection algorithm. Crossover generates one (or more) solutions using two (or more) individuals. Crossover operators for real-coded GAs are qualitatively similar to crossover operators defined for binary-coded GAs, though the non-discrete nature of the search space can be used to define new varieties of operators [7, 34]. A precise description of crossover operators is not needed for our analysis. However, I distinguish between *discrete* crossover operators, which can generate a finite number of possible solutions from a given pair of parent solutions, and *indiscrete* crossover operators, which can generate an infinite number of solutions from a given pair of parent solutions. For example, the application of uniform crossover to "swap " the coordinates of the parent solutions can generate a finite number of possible solutions. In contrast, the *blend* crossover described by Eshelman and Shaffer [7] can generate an infinite number of possible solutions along the line connecting the two parent solutions.

Mutation operates by independently perturbing each dimension of an individual probabilistically. This can be modeled by either (a) adding the value of a random variable to a dimension of an individual, called *additive mutation*, or (b) replacing a dimension of an individual with the value of a random variable, called *replacement mutation*. For example, Hart [13] describes the *interval* mutation operator, which replaces the value at a dimension of an individual with a uniformly selected value over the domain of that dimension. Similarly, Hart [13] describes a *Cauchy* mutation operator which adds the value of a Cauchy random variable to a dimension of an individual. Again, I distinguish between *discrete* mutation operators, which can generate a finite number of possible solutions a given solution, and *indiscrete* mutation operators, which can generate an infinite number of solutions from a given solution.

The replacement algorithm determines which members of the previous population are replaced by the individuals generated by the genetic operators. Typically, the entire population or a randomly selected subset of the population is replaced by the new individuals. In *elitist* EAs, the replacement algorithm ensures that the best individual in the previous population is kept in the current population if it is better than all of the newly constructed individuals.

2.2 Analyses of Evolutionary Algorithms

In this section, I review literature related to the convergence of EP, ES and real-coded GAs. These methods have been successfully applied to the optimization problems with continuous objective functions over $\mathbf{R}^n$. Traditionally, researchers have used GAs with binary-coded genes that are decoded into real values to solve this problem [5]. However, special manipulation of this encoding is often necessary to increase the efficiency of the algorithm [26, 33]. Recent research on real-coded GAs suggests that they can be more efficient, provide increased precision, and allow for genetic operators that are more appropriate for a continuous domain [7, 15, 34].

Convergence analyses of real-coded GAs have naturally focused on the role of crossover, since crossover is typically the dominant mechanism of generating new solutions in GAs. Wright [34], Goldberg [11], Eshelman and Schaffer [7], and Qi and Palmeri [20, 21] have analyzed how different types of real-coded GAs process schemata. Wright [34] defines connected schemata, which represent closed intervals along each dimension of the search domain. An analysis of these schemata shows that the schema theorem applies to real-coded GAs. Goldberg [11] uses one-dimensional slices of the search space to discuss the behavior

of real-coded GAs that use standard, coordinate-wise crossover. He argues that a GA's convergence can be blocked when the population diversity is reduced such that crossover can only generate solutions in a finite number of basins of attraction that do not contain near-optimal solutions. A similar analysis by Qi and Palmeri [21] analyzes the diversification role of coordinate-wise crossover of real-coded GAs with an infinite population size. Eshelman and Schaffer [7] discuss crossover operators for real-coded GAs that may circumvent these convergence problems by exploiting the gradualness of continuous functions. They define interval schemata, which are closely related to Wright's connected schemata, and provide a theoretic and empirical analysis of the failure modes of a particular crossover operator, the blend crossover operator.

Unfortunately, these types of schema analyses have not successfully provided provable convergence guarantees for real-coded GAs.[1] In fact, Rudolph's analysis of binary-coded GAs makes it clear that "The schema theorem [14] does not imply that the Canonical GA will converge to the global optimum in static optimization problems." [25]

Convergence analyses of EP, ES and real-coded GAs that focus on the role of mutation have been successful at providing provable convergence guarantees. Qi and Palmeri [20] provide convergence results for real-coded GAs with an infinite population size. Their results show that the distribution of solutions generated by a real-coded GA with selection alone converges in distribution to the distribution concentrated at the optimum, and that the mean fitness of the populations of a real-coded GA with selection and mutation converges to the fitness of the optimal solution. For EP, ES and real-coded GAs with finite populations, a proof of convergence to solutions with near-optimal fitness can be shown using the Borel-Cantelli Lemma [1, 2, 25, 27]. This proof requires that the mutation operator be applied with nonzero probability such that the joint distribution of possible new solutions has nonzero probability everywhere. Unfortunately, this convergence proof is too weak to provide information about the rate of convergence of these methods.

These convergence results can be improved when examined specific classes of functions. Bäck, Rudolph and Schwefel [2] have analyzed the rate of convergence of EP and ES in special contexts. They note that these methods have a geometrical convergence rate on strongly convex functions if the step size used by mutation is adapted. This result relies on a more general analysis by Rappl [22, 23].

3 Generalized Pattern Search

3.1 Description

Pattern search methods are a class of direct search methods - methods that neither require nor explicitly approximate derivatives. Torczon [31] provides an abstract generalization of pattern search methods. This abstract formulation of pattern search methods is used to establish a global first-order stationary point convergence theory that does not use the gradient or directional derivative to guarantee convergence. Here, global refers to the fact that this convergence theory guarantees convergence from any starting point. The pseudo-code in Figure 1 describes the main elements of a pattern search method.

Pattern search methods proceed by conducting a series of exploratory moves about the current iterate before identifying a new iterate. These moves can be viewed as a search about the current iterate for a trial point with a lower function value. Exploratory moves

[1] Törn and Žilinskas [32, page 78] and Rinnooy Kan [24] review the convergence guarantees typically used to define a converge theory for stochastic algorithms

1 Let the initial solution $x_0 \in \mathbf{R}^n$ and step length Δ_0 be given
2 For $k = 0, 1, \ldots$
3 a) Compute $f(x_k)$
4 b) Determine a step s_k using an *exploratory moves algorithm*
5 c) Compute $\rho_k = f(x_k) - f(x_k + s_k)$
6 d) If $\rho_k > 0$ then $x_{k+1} = x_k + s_k$. Otherwise $x_{k+1} = x_k$.
7 e) Update the *pattern matrix* and Δ_k

Figure 1: **Pattern Search**

are generated by the exploratory moves algorithm using a *pattern matrix.* The pattern matrix is decomposed into a basis matrix $B \in \mathbf{R}^{n \times n}$ and a generating matrix $C_k \in \mathbf{Z}^{n \times p}$, $p > 2n$. Given a pattern matrix BC_k, there are p possible exploratory moves $\Delta_k BC_k$, where Δ_k is a step-length parameter. Restrictions on C_k guarantee that the columns of BC_k span $\mathbf{R}^n$. Conceptually, the generating matrix defines the directions that are searched, while the basis matrix rotates and scales the search directions to determine the coordinate system used during the search.

For pattern search methods, if among the p possible exploratory moves there exists a move that provides a simple decrease in the function value, then the exploratory moves algorithm must return *some* step s_k that provides a simple decrease. The update algorithm reduces the step size if the exploratory moves algorithm fails to produce a trial step that gives a simple decrease. If the exploratory moves algorithm does produce a trial step that gives a simple decrease, then this algorithm either increases the step size or preserves the current step size.

3.2 Convergence Analysis

Torczon [31] shows that this notion of an exploratory moves algorithm can be used to prove weak first-order stationary point convergence by requiring only simple decrease on f. In particular, this proof of convergence does not explicitly require a notion of sufficient decrease on the iterates, like a fraction of Cauchy decrease or the Armijo-Goldstein-Wolfe conditions. In this section I review the major theorems in the convergence analysis of pattern search methods described in Torczon [31]. This analysis is closely related to the analysis of multidimensional search [29, 30].

The first theorem uses the simple decrease condition of the generalized pattern search algorithm, along with the algorithm for updating Δ_k, to describe the limiting behavior of Δ_k.

Theorem 1 ([31], Theorem 3.3) Assume that $L(x_0)$ is compact, then $\liminf_{k \to +\infty} \Delta_k = 0$.

Torczon uses the following proposition to prove Theorem 2.

Proposition 1 ([31], Proposition 3.4) Assume that $L(x_0)$ is compact, that f is continuously differentiable on $L(x_0)$, and that $\liminf_{k \to +\infty} |\nabla f(x_k)| \neq 0$. Then there exists a constant $\Delta_{\mathrm{LB}} > 0$ such that for all k, $\Delta_k > \Delta_{\mathrm{LB}}$.

Finally, the following theorem gives the result for weak first-order stationary point convergence for any generalized pattern search method.

Theorem 2 ([31], Theorem 3.5) Assume that $L(x_0)$ is compact and that $f : \mathrm{R}^n \rightarrow \mathrm{R}$ is continuously differentiable on $L(x_0)$. Then for the sequence of iterates $\{x_k\}$ produced by the generalized pattern search method,

$$\liminf_{k \rightarrow +\infty} |\nabla f(x_k)| = 0.$$

This convergence guarantee is weak, since it only implies that the gradient is sampled often near a stationary point. Thus it is possible for $\limsup_{k \rightarrow +\infty} |\nabla f(x_k)| > 0$. However, this convergence result is sufficient to guarantee that $\lim_{k \rightarrow +\infty} |\bar{x}^* - x_k|$ for some stationary point $\bar{x}^*$.

4 Generalized Stochastic Pattern Search

Randomness can be introduced into pattern search methods in several different ways. A simple example would be to simply shuffle the order in which the exploratory moves algorithm considers the exploratory moves. Since each move will be considered once and since there are a bounded number of exploratory moves, this randomized exploratory moves algorithm is guaranteed to return a decreasing step if one exists. Consequently, the convergence theory for pattern search methods is immediately applicable to this class of stochastic pattern search methods.

In this section, I define a different class of stochastic pattern search methods in which the exploratory moves algorithm is only probablistically guaranteed to terminate. This class of algorithms allows the generating matrix C_k^h to vary with each iteration h of the exploratory moves algorithm, provided a subset of $2n$ columns of the form $[-M_k \; M_k]$ remains fixed during each call to the exploratory moves algorithm.[2]

Because the exploratory moves algorithm is only probablistically guaranteed to terminate, it is possible that an "unlucky" sequence of exploratory moves could fail to ever provide a simple decrease. To ensure that the exploratory moves algorithm terminates with high probability, the probability of selecting each of the $2n$ fixed columns must be greater than or equal to some constant greater than zero. When all $2n$ of the search directions defined by the fixed columns have been sampled, then the exploratory moves algorithm terminates. With these restrictions, the probability is one that such an "unlucky" sequences do not occur.

4.1 Description

This section introduces the abstraction of stochastic pattern search methods, which generalizes the abstract description of pattern search methods provided by Torczon [31]. The definition of the pattern, the algorithm for generalized stochastic pattern search methods, and the update algorithm follow almost directly from Torczon's description. The definition of the exploratory moves algorithm generalizes Torczon's description to include stochastically selected exploratory moves.

[2] I use the notation $X = [YZ]$ to denote that the matrix X is partitioned into the matrices Y and Z.

The Pattern To define a pattern we need two components, a basis matrix and a generating matrix. A basis matrix can be any nonsingular matrix $B \in \mathbf{R}^{n\times n}$. A generating matrix is a matrix $C_k^h \in \mathbf{Z}^{n\times p}$, where $p > 2n$ and $h \geq 1$. We partition a generating matrix into components

$$C_k^h = [M_k \ -M_k \ L_k^h] = [\Gamma_k \ L_k^h].$$

We call Γ_k the base generating matrix. We require that $M_k \in M \subset \mathbf{Z}^{n\times n}$, where M is a finite set of nonsingular matrices, and that $L_k^h \in \mathbf{Z}^{n\times(p-2n)}$ and contains at least one column, the column of zeros.

A *pattern* P_k^h is then defined by the columns of the matrix $P_k^h = BC_k^h$. Because both B and C_k^h have rank n, the columns of P_k^h span $\mathbf{R}^n$.

Given $\Delta_k \in \mathbf{R}$, $\Delta_k > 0$, we define a *trial step* s_k^h to be any vector of the form

$$s_k^h = \Delta_k B c_k^h,$$

where c_k^h denotes a column of C_k^h. Note that Bc_k^h determines the direction of the step, while Δ_k serves as a step length parameter. At iteration k, I define a *trial point* as any point of the form $x_k^h = x_k + s_k^h$, where x_k is the current iterate.

The Exploratory Moves Pattern search methods proceed by conducting a series of *exploratory moves* about the current iterate before identifying a new iterate. These moves can be viewed as a search about the current iterate for a trial point with a lower function value. Stochastic pattern search methods differ from non-stochastic methods in that the exploratory moves algorithm is only probablistically guaranteed to terminate. Consequently, there is no fixed number of iterations of the stochastic exploratory moves algorithm; the algorithm terminates when a decreasing step has been found or after all $2n$ basic steps have been examined. In addition, the stochastic nature of the exploratory moves algorithm enables the generating matrix C_k^h to vary with each iteration h of the algorithm.

The following three conditions are placed on the exploratory moves s_k generated by an exploratory moves algorithm.[3] These three conditions constitute the **Hypothesis on Exploratory Moves.**

1. $s_k = s_k^h \in \Delta_k P_k^h$, $h = 1, 2, \ldots$
2. If $\min\{f(x_k + y), y \in \Delta_k B\Gamma_k\} < f(x_k)$, then $f(x_k + s_k) < f(x_k)$.
3. The exploratory moves algorithm terminates and returns $s_k = 0$ if each of the $2n$ *basic steps* defined by $\Delta_k B\Gamma_k$ have been (stochastically) examined without identifying a decreasing step. At each iteration of the exploratory moves algorithm, the probability of selecting each of the $2n$ basic steps is greater than or equal to a constant $\nu > 0$.

The addition of the last condition is the principal difference between this hypothesis on exploratory moves and the hypothesis on exploratory moves defined by Torczon [31]. This condition uses the observation that Torczon's convergence theory relies solely on properties of the base matrix Γ_k. Since the trial steps in $\Delta_k BL_k^h$ are not critical to the convergence of the pattern search method, it is safe to terminate the exploratory moves algorithm after all of the basic steps have been examined. The requirement that the basic steps have probability greater or equal to ν is added to ensure that the sequence of calls to the exploratory moves algorithm terminates *almost surely* (i.e. with probability one).

[3] Following Torczon, if y is a vector and A is a matrix, then $y \in A$ means that y is contained in the set of columns of A.

The Generalized Stochastic Pattern Search Method We now specify the generalized stochastic pattern search method for unconstrained minimization.

Algorithm 1 The Generalized Stochastic Pattern Search Method

Let $x_0 \in \mathbf{R}^n$ and $\Delta_0 > 0$ be given.
For $k = 0, 1, \ldots$
a) Compute $f(x_k)$.
b) Determine a step s_k using an exploratory moves algorithm.
c) Compute $\rho_k = f(x_k) - f(x_k + s_k)$.
d) If $\rho_k > 0$, then $x_{k+1} = x_k + s_k$. Otherwise $x_{k+1} = x_k$.
e) Update Γ_k and Δ_k.

To define a stochastic pattern search method, it is necessary to specify the basis matrix B, the initial values of Γ_0 and L_0^0, the exploratory moves algorithm, and the algorithms for updating Γ_k and Δ_k.

The Updates The update algorithm reduces the step size if the exploratory moves algorithm fails to produce a trial step that gives a simple decrease. If the exploratory moves algorithm does produce a trial step that gives a simple decrease, then this algorithm either increases the step size or preserves the current step size.

Algorithm 2 Updating Δ_k.

Given $\tau \in \mathbf{Q}$, let $\theta = \tau^{\omega_0}$ and $\lambda_i \in \Lambda = \{\tau^{\omega_1}, \ldots, \tau^{\omega_L}\}$, where $\tau > 1$ and $\{\omega_0, \omega_1, \ldots, \omega_L\} \subset \mathbf{Z}$, $L = |\Lambda| < \infty$, $\omega_0 < 0$, and $\omega_i \geq 0$, $i = 1, \ldots, L$.
a) If $\rho_k \leq 0$ then $\Delta_{k+1} = \theta \Delta_k$.
b) If $\rho_k > 0$ then $\Delta_{k+1} = \lambda_i \Delta_k$.

4.2 Convergence Theory

This section extends the theoretical analysis in Torczon [31] to provide convergence results for the stochastic pattern search methods. The following lemma shows that the set of sequences of trial steps for which each exploratory move terminates has probability one. This lemma enables us to apply the theoretical analysis for generalized pattern search methods to generalized stochastic pattern search methods with the confidence that the later will converge almost surely.

Lemma 1 Let A_j represent the set of sequences of trial steps generated by Algorithm 1 for which the $(j+1)$th application of the exploratory moves algorithm fails to terminate. Then $P\left(\bigcup_{j=0}^{\infty} A_j\right) = 0$.

Proof. The exploratory moves algorithm terminates when either a trial step is generated that gives a simple decrease on the function value or all $2n$ basic steps defined by $\Delta_k B \Gamma_k$ have been examined. For sequences in A_j, no trial step generated in the $(j+1)$th application of the exploratory moves algorithm provides a simple decrease, and all basic steps are never examined.

Let $A_j^i \subseteq A_j$ represent the subset of sequences in A_j for which the ith basic step is never examined in the $(j+1)$th application of the exploratory moves algorithm. Thus $A_j = \bigcup_{i=1}^{2n} A_j^i$. Let $A_j^{i,m}$ be the subset of A_j^i for which the i-th basic step is not examined in

the first m trial steps of the $(j+1)$th application of the exploratory moves algorithm. Since each of the $2n$ basic steps has probability ν of being selected, $P\left(A_j^{i,m}\right) \leq (1-\nu)^m$. Now $A_j^i = \bigcup_{m=1}^{\infty} A_j^{i,m}$ and $A_j^{i,m} \subset A_j^{i,m+1}$, so $P(A_j^i) = \lim_{m\to\infty} P(A_j^{i,m}) = \lim_{m\to\infty}(1-\nu)^m = 0$. Thus

$$P\left(\bigcup_{j=0}^{\infty} A_j\right) \leq \sum_{j=0}^{\infty} P(A_j) \leq \sum_{j=0}^{\infty}\sum_{i=1}^{2n} P\left(A_j^i\right) = 0.$$

■

Note that

$$\left(\bigcup_{j=0}^{\infty} A_j\right)^c$$

is the set of sequences for which each exploratory move terminates. Thus the following corollary follows immediately from Lemma 1.

Corollary 1 The set of sequences for which each exploratory move terminates has measure one.

Using Lemma 1, the following theorem describes the limiting behavior of Δ_k that occurs almost surely.

Theorem 3 Assume that $L(x_0)$ is compact. Then $P(\liminf_{x\to\infty} \Delta_k = 0) = 1$.

Proof. If each exploratory move terminates, then we can apply Theorem 1 to show that $\liminf_{x\to\infty} \Delta_k = 0$. From Lemma 1 we know that the set of sequences for which each exploratory move terminates has measure one, so $P(\liminf_{x\to\infty} \Delta_k = 0) = 1$. ■

The following theorem uses the limiting behavior of Δ_k to extend the first-order stationary point convergence to stochastic pattern search methods. This is a global convergence theory since it is applicable for any initial point x_0 in the search domain, and not simply for an initial point that is sufficiently close to the stationary point.

Theorem 4 Assume that $L(x_0)$ is compact and that $f : \mathbf{R}^n \to \mathbf{R}$ is continuously differentiable on $L(x_0)$. Then for the sequence of iterates $\{x_k\}$ produced by the stochastic generalized pattern search method (Algorithm 1),

$$P\left(\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0\right) = 1.$$

Proof. The proof is by contradiction. Suppose that $P(\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0) < 1$. Then there exists a set of sequences of iterates $\{x_k\}$ with measure greater than zero for which $\liminf_{k\to\infty} \|\nabla f(x_k)\| \neq 0$. For these sequences, we know from Proposition 1 that there exists $\Delta_{LB} > 0$ such that $\Delta_k > \Delta_{LB}$. But this contradicts Theorem 3. ■

5 Evolutionary Pattern Search Algorithms

Now I define a class of EAs that can be described as stochastic pattern search methods, which I refer to as EPSAs. The following section describes a variation of the canonical EA that uses discrete crossover and mutation operators. The next section examines the rate of convergence of EPSAs, and evaluates the effect of the rate of crossover on the rate of convergence. The final section discusses stopping rules for EPSAs.

5.1 Description

The crossover operator generates individuals using coordinate-wise swaps, and the mutation operator adds a discrete random variable to a dimension of an individual, which may assume values $\{\sigma_1, \ldots, \sigma_m\}$, $\sigma_i \in \mathbf{Q}$. Let $\sigma_{min} = \min_{i=1,\ldots,m} |\sigma_i|$. Let e_i be the standard unit vector in the i-th dimension. Let $\eta \in \{0,1\}^{2n}$ be a counter that indicates whether the $2n$ basic vectors have been examined. The initial step length is $\Delta_0 \in \mathbf{R}^+$, and the contraction factor $\theta \in \mathbf{Q}$, $0 < \theta < 1$. Figure 2 shows the pseudo code describing EPSAs.

```
1  Initialize the population of individuals {x_1^0, ..., x_N^0} such that x_i^0 ∈ Q^n
2  Compute the fitness of each individual and reorder such that x_1^t ≤ x_i^t, ∀i
3  Let t = 0 and η = 0^{2n}
4  Repeat k = 0, 1, ...
5     Repeat h = 1, 2, ...
6        Perform selection with a policy which guarantees that the
            individual with the smallest fitness value is selected for processing
            at each generation with a probability of at least π > 0
7        Perform crossover with probability χ ∈ [0, 1)
8        Perform mutation with probability μ ∈ (0, 1] of mutating each dimension; mutation adds
            a vector of the form Δ_k ω, ω ∈ {0, ±σ_1, ..., ±σ_m}^n, to an individual
9        If the mutation x_1^t + Δ_k γ σ_min e_i is generated, then η_i = 1
         Else, if the mutation x_1^t − Δ_k γ σ_min e_i is generated, then η_{i+n} = 1
10       Compute the fitness of each individual and reorder such that x_1^t ≤ x_i^t, ∀i
11       Perform replacement with an elitist replacement policy which guarantees that
            the individual with the smallest fitness value among the previous population
            and the newly generated individuals is included in the next generation.
12       t = t + 1; x_t^* = arg min_{i=1,...,N} f(x_i^k) and y_t^* = f(x_t^*)
13    Until (y_t^* < y_{t-1}^*) or (Σ_{i=1}^{2n} η_i = 2n)
14    If y_t^* = y_{t-1}^* then Δ_{k+1} = θ Δ_k
15    η = 0^{2n}
16 Until some stopping criterion is satisfied
```

Figure 2: **Pseudo Code for EPSAs**

The inner loop in the EPSA is used to highlight the relationship between the EPSA and generalized stochastic pattern search methods. This loop can easily be merged with the outer loop to provide a formulation of the EPSA that more closely resembles canonical EAs. The restriction on the replacement strategy ensures that the best individual found is kept for further processing. Together, these restrictions enable the EPSA to be viewed as a pattern search method with respect to the best individual in the each generation.

Note that EPSAs manipulate solutions that are vectors in $\mathbf{Q}^n$. This is necessary in order to describe the EA as a generalized pattern search method. In principle, this distinguishes EPSAs from the canonical EAs, since these EAs manipulate vectors in $\mathbf{R}^n$. However, this distinction is not of practical importance. The experimental evidence in Section 6 confirms that the use of rational vectors does not impede an EPSA's ability to optimize functions defined over $\mathbf{R}^n$.

In what follows I describe how the three central components of stochastic pattern search

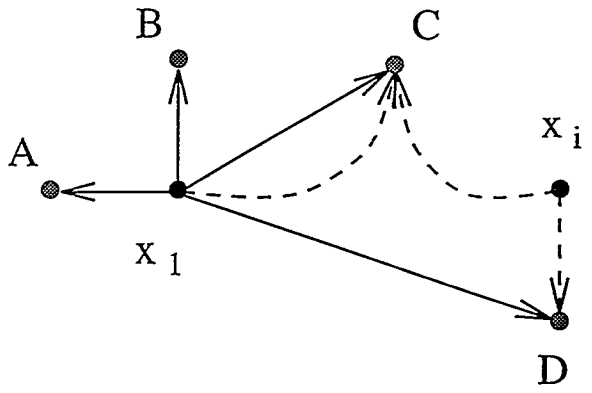

Figure 3: Illustration of pattern vectors that are offsets from the best individual, x_1.

– updating step length, the generating matrix, and the exploratory moves algorithm – are implemented by the EPSA.

Updating the Step Length Step 14 of the EPSA performs the update to the step length Δ_k. This update is exactly as given in Algorithm 2. When the inner loop fails to generate a simple decrease on f, the step length is decreased by a multiple of θ. The settings that are defined by the EPSA are $\theta = 1/2$ and $\Lambda = \{1\}$. Thus $\tau = 2$, $\omega_0 = -1$ and $\omega_1 = 0$.

The Matrices Pattern search methods are designed to generated test patterns from a single solution. To cast the EPSA within this framework, the individuals generated at each generation are viewed as patterns with respect to the best individual in the population. Figure 3 illustrates the pattern vectors generated by individuals created by crossover and mutation. Points A and B represent individuals created by mutation from the best individual, x_1. Point C represents an individual created by crossover from x_1 and x_i. Point D represents an individual created by mutation of x_i. The arrows represent the pattern vectors, which are implicitly scaled by the basis matrix to lie in $\mathbf{Z}^n$.

The generating matrix C_k^h is constructed using all possible individuals that can be generated in the current generation t (as indexed by k and h). Let $f(x_1^t) \leq f(x_j^t)$, $j = 2, \ldots, N$, for all t. The columns of C_k^h are generated by taking each individual $\hat{x}$ that can be generated by the EPSA and constructing the vector $\Delta_0\gamma(\hat{x} - x_1^t)/\Delta_k$, where $\gamma = \gcd(x_{i,j}^0, \sigma_i)$.[4] The basis matrix implicitly used by the EPSA is $B = \frac{1}{\Delta_0\gamma}I$, where I is the identity matrix, so the trial steps are

$$\Delta_k B c_k^h = \Delta_k B \Delta_0 \gamma(\hat{x} - x_1^t)/\Delta_k = (\hat{x} - x_1^t),$$

It remains to prove that the patterns vectors c_k^h are in $\mathbf{Z}^n$. To prove this, we recall the following theorem.

Theorem 5 ([31], Theorem 3.2) Any iterate x_N produced by a generalized search method

[4]The function $\gcd(q_0, \ldots, q_k)$ returns the greatest common divisor from among the quotients q_i.

can be expressed in the following form:

$$x_N = x_0 + (\beta^{r_{LB}}\alpha^{-r_{UB}})\,\Delta_0 B \sum_{i=0}^{N-1} z_k,$$

where

- x_0 is the initial iterate
- $\beta/\alpha = \tau$, with $\alpha, \beta \in \mathrm{Z}^+$ and relatively prime, and τ is as defined in the algorithm for updating Δ_k
- $r_{LB} = \min_{0\le k\le N} r_k$ and $r_{UB} = \max_{0\le k\le N} r_k$, where $\Delta_k = \tau^{r_k}\Delta_0$
- $z_k \in \mathrm{Z}^n$, $k = 0, \ldots, N-1$.

The following corollary is a straightforward generalization of this theorem for the EPSA. The fact that $r_{LB} \le r_{UB} \le 0$ is derived from the fact that Δ_k is a nonincreasing sequence.

Corollary 2 Suppose the pattern matrices C_k^h for $t = 0, \ldots, T-1$ are such that $C_k^h \in \mathrm{Z}^{n\mathrm{X}p}$. Then any individual x_i^T produced by an EPSA can be expressed in the following form:

$$x_i^T = x_1^0 + (\beta^{r_{LB}}\alpha^{-r_{UB}})\,\Delta_0 B \sum_{k=0}^{T-1} z_k.$$

where $r_{LB} \le r_{UB} \le 0$.

Using this corollary, the following lemma proves that the pattern vectors we have defined are in Z^n. The corollary requires that the pattern matrices used to generate x_i^t be in $\mathrm{Z}^{n\mathrm{X}p}$. Since this is what we are trying to prove, the proof of this lemma is by induction on t.

Lemma 2 $C_k^h \in \mathrm{Z}$ for all $h = 0, 1, \ldots$ and $k = 0, 1, \ldots$.

Proof. The proof is by induction on the time step T. If $T = 0$, then a pattern has the form

$$\Delta_0\gamma(\hat{x} - x_1^T)/\Delta_k = \gamma(\hat{x} - x_1^T) \in \mathrm{Z}^n.$$

Now suppose that $C_k^h \in \mathrm{Z}^n$ for $t = 0, \ldots, T-1$. We consider the case when an individual $\bar{x}$ is generated by the crossover; a similar argument follows for mutation without crossover. The new individual can be expressed as

$$\bar{x} = x_1^0 + (\beta^{r_{LB}}\alpha^{-r_{UB}})\,\Delta_0 B \left(\sum_{k=0}^{T-1} I_1 z_k + \sum_{k=0}^{T-1} I_2 z_k \right),$$

where I_1 and I_2 are matrices that account for the mapping performed by crossover. An individual generated by crossover and mutation can be expressed as $\bar{x} + \Delta_k B\omega$. Thus the pattern at time T has the form

$$\begin{aligned} \Delta_0\gamma(\hat{x} - x_1^t)/\Delta_k &= \Delta_0\gamma(\bar{x} + \Delta_k B\omega - x_1^i)/\Delta_k \\ &= \gamma\omega + (\beta^{r_{LB}}\alpha^{-r_{UB}})\,\frac{\Delta_0}{\Delta_k}\sum_{k=0}^{t-1} z_k' \end{aligned}$$

for some vectors $z'_k \in \mathbf{Z}^n$. From the definition of γ, we know that $\gamma\omega \in \mathbf{Z}^n$. New $\Delta_k = (\beta/\alpha)^{r_k}\Delta_0$, $r_k \leq 0$. Thus

$$(\beta^{r_{LB}}\alpha^{-r_{UB}})\frac{\Delta_0}{\Delta_k}\sum_{k=0}^{t-1} z'_k = (\beta^{r_{LB}-r_k}\alpha^{r_k-r_{UB}})\sum_{k=0}^{t-1} z'_k.$$

Since the exponents for β and α are positive, this term is also in $\mathbf{Z}^n$.

■

Finally, note that the base generating matrix is $\Gamma_k = \gamma\sigma_{min}[I \ -I]$, which is constant for all generations of the EPSA.

Exploratory Moves The exploratory moves algorithm used by the EPSA is contained within the inner loop on steps 5 through 13. As required by the exploratory moves hypothesis, this loop only terminates if a solution is found that generates a simple increase, or if all $2n$ basic steps defined by $\Delta_k B\Gamma_k$ have been examined.

All possible individuals that can be generated by the EPSA are captured in patterns in C_k^h. Suppose that the discrete random variable used to perform mutation generates a values $\pm\sigma_i$ with probability η each. The base generating matrix represents the single-coordinate mutations for which mutation occurs with values σ_{min}. The restriction on the selection strategy ensures that each of these single-coordinate mutations can occur on x_1^t with probability greater than or equal to $\delta = \pi\eta\mu(1-\mu)^{n-1}$. Thus, the exploratory moves algorithm implicitly defined by the EPSA satisfies the hypothesis on exploratory moves.

5.2 Convergence Analysis

The previous section demonstrated that EPSAs can be described as stochastic pattern search methods. Consequently, from Theorem 1 we know that

$$P\left(\liminf_{t\to\infty} \|\nabla f(x_t^*)\| = 0\right) = 1.$$

This analysis guarantees that convergence exists, but does not indicate the *rate* at which stochastic pattern search algorithms converge. However, it has long been recognized that pattern search methods do not enjoy fast local convergence properties [31]. We defer a discussion of the *rate* of convergence of EPSAs (and pattern search algorithms in general) until Section 8. In this section, we examine the relative computational costs of EPSAs with different genetic operators.

The following lemma quantifies the expected length of time spent by the exploratory moves algorithm. Recall from our definition of stochastic pattern search, the probability of selecting each of the $2n$ trial steps is greater than or equal to a constant $\nu > 0$. The following lemma provides an upper bound on the expected time spent in the stochastic exploratory moves algorithm.

Lemma 3 Let Y_n be the number of iterations spent in the stochastic exploratory moves algorithm. Then

$$E(Y_n) \leq \frac{1}{\nu}\sum_{k=1}^{2n}\frac{1}{k} = \frac{1}{\nu}O(logn).$$

Proof. Consider each iteration of the exploratory moves algorithm as a sample with replacement from a set of exploratory moves. In the worst case, the exploratory moves algorithm termimates when all of the basice steps have been sampled.

Let $S_{k,2n}$ be the number of samples until k of the $2n$ elements are sampled, and let $X_{k,2n} = S_{k,2n} - S_{k-1,2n}$. $X_{k,2n} - 1$ is the number of elements sampled between the $(k-1)$th new basic step and the kth new basic step. The value $X_{k,2n} - 1$ is geometrically distributed with parameter $p \geq \nu(2n - k + 1)$. Thus

$$E(X_{k,2n}) = 1 + \frac{1-p}{p} = \frac{1}{p} \leq \frac{1}{\nu(2n-k+1)}.$$

Now $S_{2n,2n}$ is the number of samples required to ensure that all $2n$ trial steps have been sampled. Since

$$S_{2n,2n} = X_{2n,2n} + \ldots + X_{1,2n},$$

we have

$$E(Y_n) = E(S_{2n,2n}) \leq \frac{1}{\nu} \sum_{k=1}^{2n} \frac{1}{(2n-k+1)}.$$

Now note that

$$\sum_{k=1}^{2n} \frac{1}{2n-k+1} = \sum_{k=1}^{2n} \frac{1}{k} \leq \log(2n+1).$$

Thus $E(Y_n) \leq \frac{1}{\nu} O(logn)$. ∎

Since EPSAs generate a fixed number of possible offspring at each generation, this lemma is applicable. In general, it is not possible to determine the value of ν, even if the mutation and crossover rates are provided. The reason is that selection can greatly influence the probability that the best individual in the population is selected. However, for rank selection the value of ν is easily computed.

Let r_i be the rank of the ith individual in the population, such that the best rank is 0 and the worst rank is $P-1$, where P is the size of the population. Linear ranking [12] assigns a probability of selection to an individual x_i according to the equation

$$p(x_i) = \frac{C - 2(C-1)r_i/(P-1)}{P},$$

where $1 \leq C \leq 2$. It follows that the probability of selecting the best individual in a population is at least C/P (it could be greater if multiple copies of the best individual exist).

I consider the effects of two types of discrete mutation operators. The *binomial* mutation operator is the analogue of the mutation operator commonly used in binary GAs. The probability of mutation a given dimension of x_i by σ_j is $\mu/(2m)$. Similarly, the probability of mutation a given dimension of x_i by $-\sigma_j$ is also $\mu/(2m)$. The *multinomial* mutation operator mutates a single dimension of x_i with probability μ. Multinomial mutation is introduced here because it has a much higher probability of generating singleton mutations that are used by EPSAs to control the step length parameter. Given that mutation occurs, the probability of mutating by σ_j is $1/(2m)$.

For EPSAs using binomial mutation, the probability of generating each of the basic steps is C/P times the probability that mutation generates a basic step and crossover is not applied. The probability that binomial mutation generates a basic step is

$$\mu(1-\mu)^{n-1}/(2m).$$

Thus the probability that EPSAs using binomial mutation generate a basic step is

$$\nu \geq \frac{C(1-\chi)\mu(1-\mu)^{n-1}}{2mP}.$$

The probability that multinomial mutation generates a basic step is

$$\mu/(2nm).$$

Thus the probability that EPSAs using multinomial mutation generate each of the basic steps is

$$\nu \geq \frac{C(1-\chi)\mu}{2nmP}.$$

The following corollary applies Lemma 3 to bound the length of the exploratory moves for EPSAs that use linear rank selection with binomial or multinomial mutation.

Corollary 3 Let an EPSA use linear rank selection with parameter C. If binomial mutation is used, then

$$E(Y_n) \leq \frac{2mP}{C(1-\chi)\mu(1-\mu)^{n-1}} \sum_{k=1}^{2n} \frac{1}{k} = \frac{2mP}{C(1-\chi)\mu(1-\mu)^{n-1}} O(\log n).$$

If multinomial mutation is used, then

$$E(Y_n) \leq \frac{2nmP}{C(1-\chi)\mu} \sum_{k=1}^{2n} \frac{1}{k} = \frac{2mP}{C(1-\chi)\mu} O(n \log n).$$

Let μ_b be the mutation rate used for binomial mutation, and let μ_m be the mutation rate used for multinomial mutation. The length of the exploratory moves algorithm to be shorter for EPSAs using binomial mutation if

$$\frac{1}{\mu_b(1-\mu_b)^{n-1}} < \frac{n}{\mu_m}.$$

If $\mu_b = \mu_m$, this implies that

$$\mu_m = \mu_b < 1 - n^{-1/(n-1)}.$$

Even for moderately large n, the expected length of the exploratory moves algorithm for EPSAs using binomial mutation will be shorter. Intuitively, the reason for this is that the expected number of mutations per individual is $\mu_b n$ for binomial mutation but only μ_m for multinomial mutation. If we equalize the expected number of mutations, by setting $\mu_m = n\mu_b$, then the expected length of the exploratory moves algorithm will *always* be shorter for EPSAs using multinomial mutation.

5.3 Stopping Rules

The proof of convergence for EPSAs relies soley on the properties of the mutation operator. However, the constrained nature of the exploratory moves should make stopping rules for EPSAs applicable in practice. Torczon [29] discusses a number of stopping rules for direct search methods that might be applicable to EPSAs. These stopping rules cannot

guarantee that the algorithm is terminated at a stationary point because the gradient is not available and is not explicitly estimated. Instead, these stopping rules measure whether the algorithm has "ground to a halt," either because it has stalled or converged.

The stopping rules discussed by Torczon [29] are applied to direct search methods for which all points in the simplex of possible new points are examined. In generalized pattern search methods, the stopping rule is checked when the exploratory moves algorithm fails to find an improving search direction. At this point, we are only guaranteed to have examined the basic steps. Consequently, the simplex used to evaluate the stopping rule for pattern search methods is simply the simplex of basic steps.

Let $v_1^t, \ldots, v_{2n}^t$ be the $2n$ basic steps in a pattern search algorithm at time t, such that $f(v_1^t) \leq f(v_i^t)$ for $i = 1, \ldots, 2n$. The stopping rule recommended by Torczon [29] terminates the algorithm when

$$\frac{1}{\delta} \max_{1 \leq i \leq 2n} \left| v_i^t - v_1^t \right| \leq \epsilon,$$

where $\delta = \max(\xi, |v_1^t|)$. The division by δ makes this stopping rule terminate based on the relative change the step lengths [6]. This type of stopping rule may be satisfactory if the change in step lengths is greater than ξ, but will be unsatisfactory if the change in step lengths is always much smaller than ξ. Dennis and Schnabel [6] suggest that a reasonable guideline for choosing the value of the step tolerance ϵ: if p significant digits are desired, then set ϵ to 10^{-p}.

For EPSAs, the structure of the set of basic steps makes this stopping rule equivalent to

$$\frac{2\Delta_k}{\max(\xi, |v_1^t|)} \leq \epsilon,$$

which is essentially a stopping rule based on the value the step length Δ_k.

6 Experimental Evaluation

The experiments described in this section evaluate the performance of EPSAs. These exepriments address a number of practical questions concerning EPSA.

- How important is an adaptive step size? If EAs using fixed step lengths perform as well as EPSAs, then there may be little practical need for the machinery needed to adapt the step length.

- What is the effect of the rate of crossover on the total cost of EPSAs? Our analysis in Section 5.2 enables us to predict the relative cost of each iteration of EPSAs, which is one factor of the total cost. Other factors such as the rate of convergence are related to the role of crossover in EAs. This question also addresses the ability of the stopping rule to terminate in the presence of crossover steps. Since the stopping rule depends upon the fitness landscape with respect to mutation steps, it is not clear whether or not the stopping rule would become less useful for EPSAs using crossover.

- How does the performance of EPSAs compare with that of canonical EAs? Specifically what is the effect of using a discretized mutation operator in EAs?

- How well do the stopping rules work? Do they terminate EPSAs near critical points of the search space?

Two functions were used in these experiments. The first function is the modified Rastrigin function described by Mühlenbein, Schomisch and Born [18]. This function has the form

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

which generalizes Rastrigin's function to an arbitrary number of dimensions. Following Mühlenbein et al., we optimize this function in 20 dimensions, over the domain $[-5.12, 5.12]^{20}$. The local minima of this function are approximately located on the integer coordinates, and the global minimum is at the origin. In the experiments minimizing this function, the search is terminated if a solution with a value below 10^{-3} is found, since these solutions are known to lie in the basin of attraction of the global minimum [18].

The second function is the two-dimensional molecular conformation problem described by Judson et al. [16]. This problem concerns a molecule composed of a chain of identical atoms that are connected with rigid rods of length one. The energy of the molecule is modeled van der Walls forces. The energy equation used by Judson et al. is

$$f(\alpha) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \left[\left(\frac{1}{r_{ij}} \right)^{12} - 2 \left(\frac{1}{r_{ij}} \right)^{6} \right]$$

, where $r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. The angles α represent the bond angles for a molecule, which can be used to calculate the coordinates x_i and y_i (see Hart [13]). A molecule with 19 atoms was used in our experiments, which requires 17 bond angles to specify its conformational space. The optimal conformation has an energy of approximately -45.3.

The canonical EAs used in the experiments are equivalent to the real-coded GAs described by Hart [13]. A binomial additive mutation operator is used that adds a normally distributed random deviate to the dimensions of an individual that are mutated.

The EPSA used in the experiments initializes individuals by discretizing the range of each dimension into 100 values. The mutation operator adds a value of Δ_k or $-\Delta_k$ to a dimension of an individual. Note that the greatest common divisor γ does not need to be explicitly computed to execute the EPSA. This value is included in the formalism to clarify the relationship between EPSAs and stochastic pattern search methods. The initial step length used in the experiments is $\Delta_0 = 1$. The contraction factor is $\theta = 1/2$. The value $\xi = 10^{-8}$ is used by the stopping rule. and the threshold is $\epsilon = 10^{-4}$.

In all experiments, the EPSA and EAs use a population size of 100 and results are averaged over 20 different seeds for the random number generator. Linear rank selection is used to select individuals for recombination and mutation. Elitist replacement is used to ensure that the best individual seen is kept in subsequent populations. Efforts were made to reduce the total number of function evaluations by keeping track of individuals that were generated from identical parents and individuals that were simply copied to the next population (e.g. the elitist individual).

6.1 Effects of Adaptive Step Sizes

This experiment compares the performance of the EPSA to the performance of EAs that use a fixed step size. Step sizes of 1.0. 0.1, 0.01, and 0.001 were used. Table 1 summarizes the final performance of the EPSA and EAs. The EPSA was terminated using the stopping rule.

Δ	Best	Mean	Worst
Rastrigin			
EPSA	0.0009	3.4327	7.9597
1e-0	92.8243	142.8100	182.3980
1e-1	64.3360	124.3140	175.9030
1e-2	60.7176	121.7640	174.1580
1e-3	93.8594	143.0180	186.1240
Judson			
EPSA	0.0009	3.4327	7.9597
1e-0	-35.6240	-33.3723	-30.8315
1e-1	-42.1881	340.6510	1526.1300
1e-2	-43.2082	397.8890	2245.0900
1e-3	-39.5729	463.4200	2745.2500

Table 1: Comparison of performance for the EPSA with adaptive step length against EPSAs with fixed step length Δ.

The EAs were terminated after 400,000 and 650,000 function evaluations for the Rastrigin and Judson functions respectively, which is an upper bound on the maximum number of function evaluations required by the EPSA for these problems. These results indicate that there is a best fixed step size for the EAs. However, it is clear that the adaptation of step sizes by the EPSA allows it to find much better solutions.

6.2 Effects of Crossover

Our analysis in Section 5.2 enables us to predict the relative cost of each iteration of EPSAs, which is one factor of the total cost. Specifically, this analysis predicts that in the worst case, a crossover rate of χ will multiply the number of generations used by the EPSA by a factor of $1/(1-\chi)$. This analysis does not incorporate the effects that crossover may have on the convergence rate of the EPSA, since it is a worst case analysis.

Table 2 compares the performance of the EPSA when crossover is not used $\chi = 0.0$, and when crossover is used frequently, $\chi = 0.8$. For this experiment, the analysis predicts that the EPSA using crossover will require on average five times as many generations (calls to the exploratory moves algorithm) as the EPSA that does not use crossover. For the Rastrigin function, the ratio between the average number of function evaluations used by the EPSAs is remarkably close to this prediction, which suggests that introducing crossover does not significant affect the rate of convergence of EPSAs for this problem. For the Judson function, the ratio is smaller, which suggests that crossover does somehow increase the rate of convergence. In both cases, the use of crossover improves the average performance of the EPSAs. The use of a stopping rule to terminate the EPSAs clarifies the relationship between the time spent searching and the quality of the solution found. The EPSAs using crossover find better solutions at the expense of additional function evaluations.

6.3 EPSAs vs canonical EAs

This experiment compares the performance of the EPSA with and without crossover to the performance of the canonical EA, with and without crossover. The EPSAs and

χ	Best	Mean	Worst	Avg Neval
Rastrigin				
0.0	0.0009	3.4327	7.9597	342641
0.8	0.0009	0.7964	3.9798	1479460
Judson				
0.0	-44.2142	-42.1958	-39.6232	646571
0.8	-43.2465	-42.2229	-40.0735	2199190

Table 2: Comparison of performance for EPSAs when $\chi = 0.0$ and $\lambda = 0.8$.

	χ	Best	Mean	Worst
Rastrigin				
EPSA	0.0	0.0009	3.4327	7.9597
	0.8	1.0513	5.8319	23.2171
EA	0.0	0.0627	0.2213	0.3945
	0.8	0.1090	0.2335	0.4073
Judson				
EPSA	0.0	-44.2142	-42.1958	-39.6232
	0.8	-43.0146	-41.3107	-39.3882
EA	0.0	-44.2740	-39.8110	-36.4797
	0.8	-42.5970	-40.5651	-38.6333

Table 3: Comparison of performance for EPSAs with EAs.

EAs were terminated after 400,000 and 650,000 function evaluations for the Rastrigin and Judson functions respectively. Table 3 summarizes the results of this experiment.

These results indicate that the relative performance of EAs and EPSAs depends upon the objective function that is minimized. The EAs outperform the EPSAs on the Rastrigin function, but the opposite is true for the Judson function. To understand this difference, consider the following difference in the two functions. The Judson function has narrow local minima with steep sides, so local refinement is necessary to determine the whether a particular solution in this problem is close to a near-optimal minima [16]. The Rastrigin function has much broader local minima, so the relative fitnesses of solutions provide a reasonable measure of their relative distances to near-optimal solutions.

Thus it appears that EPSAs and EAs will prove best at solving different classes of functions. The local refinement performed by EPSAs should be useful on functions for which the measures of the sets of near-optimal solutions around the local minima are small. The broad mutation operator used by EAS should be useful on functions for which the mutation operator can improve the solution by moving to better local minima.

6.4 Efficacy of the Stopping Rule

To evaluate the efficacy the stopping rules, I refined the final solutions from the optimization trials in the experiment from Section 6.2. The final solutions were minimized using conjugate gradient [19], a gradient-based optimization method. The refined solutions were compared to the final solutions generated by the EPSA. In all cases, the final solutions generated by the EPSA were within the epsilon tolerance (10^{-4}) of the refined solutions.

This result confirms that the stopping rule can effectively terminate EPSAs near stationary points.

7 Extensions

This section discusses a number of extensions of the basic formulation and analysis of EPSAs described in Section 5.

7.1 Nondifferentiable Objective Functions

A strength of EAs is their ability to minimize nondifferentiable functions on which standard optimization methods are not effective. The convergence theory for EPSAs appears to sacrifice this advantage, since it is applicable to differentiable functions. In fact, the convergence theory for pattern search algorithms can be generalized to continuous functions [31, 30]. This generalized convergence theory guarantees that EPSAs converge almost surely to stationary points, as well as points where the function is nondifferentiable or where the gradient exists but is not continuous. For example, for the function

$$f(\bar{x}) = x_1 + |x_1| \sum_{i=1}^{n} x_i^2,$$

EPSAs will converge to some solution for which $x_1 = 0$.

7.2 Increasing the Step Length

Direct search methods that adapt a step length typically include rules for both reducing and increasing the step length. While this is true for pattern search methods as well, the EPSA defined in Section 5.1 only reduces the step length. The analysis in Section 5.1 that demonstrates that EPSAs are stochastic pattern search methods only applies when the step length is nonincreasing. If the crossover rate is zero, then it is possible to show that EPSAs that reduce and increase the step length also converge. The proof of this result is similar to the presentation in Section 5.1, and simply uses a different generating matrix.

I examined an EPSA using a nonzero crossover rate that doubled the step length every time an improving step is made to evaluate the effects of using increasing step lengths with crossover. This rule is similar to that used by a number of other pattern search methods. Unfortunately, this rule quickly led to very large step lengths. The reason is that during early generations, the crossover operator is the principal means by which EPSAs generate improved solutions. Consequently, the increases in step length bear little relation to the improvement exhibited by an EPSA. This is problematic because when crossover begins to produce improvements less frequently, the step length is so large that the EPSA spends a great deal of effort simply reducing its step length to something reasonable. This phenomenon was less dramatic in experiments for which a smaller expansion factor was used, which suggests that it may be possible to develop EPSAs that using increasing step lengths with crossover.

7.3 Constraints

While Torczon [31] addresses convergence for unconstrained optimization problems, I expect that these methods will be applicable to constrained optimization problems as well.

The convergence theory can be easily adapted to handle constraints by ignoring solutions that violate the constraints. Thus the exploratory moves algorithm is guaranteed to return an improving step if and only if the set of *feasible* basic steps includes an improving solution. Sophisticated methods of constrained optimization will certainly be needed ensure that the number of infeasible solutions generateed by EPSAs is not excessive, but the basic analysis of the convergence theory should be adequate to guarantee the convergence of methods for constrained optimization.

7.4 Alternate Basis Matrices

The definition of (stochastic) pattern search methods includes the selection of a basis matrix B. The basis matrix tailors the pattern search algorithm to particular problems by modifying the coordinate system that the algorithm searches. For example, if variables are known to differ by several orders of magnitude, then this can be taken into account by the appropriate choice of the basis matrix. The EPSA described in Section 5.1 implicitly uses the basis matrix $B = 1/(\gamma\Delta_0)$. If an alternate basis matrix is chosen, then all trial steps are modified, including steps from individuals generated by crossover.

The use of a different basis matrix, especially a nondiagonal matrix, modifies the processing of the EPSA in a manner unlike that of any other type of EA. This transformation of the EPSAs search is a novel generalization of the search performed by EAs that can be investigated independent of the EPSA itself. The notion of a basis matrix combines the effects of the scale of each dimension as well as the rotation of the search space. Previously, researchers have examined problems for which these transformations were shown to provide measurable changes in the performances of GAs []. The notion of a basis matrix provides a unifying formalism for incorporating *prior* knowledge of a problem into the EA used to for optimization.

7.5 Parallel EPSAs

The convergence theory for sequential EPSAs can be extended to related parallel EAs. For example, consider an island model EA for which each processor maintains an independent population that is evolved as an EPSA and individuals are migrated between processors on occasion [18, 28]. This is a natural generalization of the sequential model of EPSAs. The convergence theory for stochastic pattern search algorithms is not immediately applicable to this algorithm because each population can independently adapt their step sizes. However, if we assume that each population begins with the same step size and reduces the step size by the same fraction, then the combined populations can be shown to lie on a common translated integer lattice. This is an important property of pattern search methods, and it enables us to conclude that the convergence theory applies to this island model EA.

8 Conclusions

The central contribution of this work is the development of a convergence theory for EAs that guarantees convergence almost surely to a critical point for any differentiable function. This convergence result is qualitatively different from proofs of convergence to a global optimum. The difference stems from the fact that the problem of finding a solution x^* for which $f(x^*)$ is the global optimum is ill-conditioned [32]. Consequently, these proofs of convergence to the global optimum only guarantee that the estimate of the fitness at

the optimum, f^t converges to $f(x^*)$. Because the convergence theory for EPSAs only guarantees convergence to *some* critical point $\bar{x}^*$, it can guarantee that the sequence of improving solutions x^t converges to $\bar{x}^*$.

One potential advantage of this type of convergence theory is that reasonable rates of convergence can be guaranteed. Rates of converegence have been difficult to establish for EAs, except for certain classes of convex functions [2]. Unfortunately, it has been difficult to establish convergence rates for EPSAs. The reason is that the convergence theory only requires a simple decrease in the objective function. In general, this is not sufficient to guarantee even a geometric rate of convergence, which is one of the weakest convergence rates typically analyzed. It may be possible to determine rates of convergence for EPSAs for specific classes of functions, like the strongly convex functions examined by Rappl [22, 23].

Torczon [31] describes a stronger convergence theory that guarantees that pattern search methods converge such that

$$\lim_{k \to +\infty} |\nabla f(x_k)| = 0.$$

I expect that convergence rates will be easier to determine for algorithms that satisfy the conditions of this convergence theory, since these conditions tie the decrease to the norm of the gradient. Among these restrictions is a bound on the norm of the pattern matrix C_t^h. For EAs, this implies that solutions would need to be culled from the population if they are too far away from the best solution in the population. The experiments for EPSAs indicate that the amount of global search they perform is reasonably close to the search performed by canonical EAs, but I expect that this type of culling would limit the global search performed by EPSAs.

Two other contributions of this work are also noteworthy. First, the convergence theory provides a method of adapting the step length of the mutation operator. Previously, methods for adapting the mutation operator have been analyzed for specific classes of functions (e.g. see Rappl [23]). However, the method of adapting the mutation operator used in EPSAs is the first such method that can be tied to the convergence of an EA for *any* continuous function.

The second other contribution is the definition of a stopping rule for EPSAs. To my knowledge, this is the first stopping rule for any EAs that can successfully terminate near a stationary point. While this rule does not guarantee that a stationary point is acheived (since it does not utilize gradient information), the experimental evidence that I have presented confirms that this stopping rule can effectively terminate EPSAs close to a stationary point.

Acknowledgements

I wish to thank John DeLaurentis for his help with the analysis of stochastic pattern search methods. I also thank Virginia Torczon, Bruce Hendrickson and Juan Meza for their helpful discussions concerning pattern search methods.

References

[1] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, *A survey of evolution strategies*, in Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, R. K. Belew and L. B. Booker, eds., San Mateo, CA, 1991, Morgan-Kaufmann, pp. 2–9.

[2] T. BÄCK, G. RUDOLPH, AND H.-P. SCHWEFEL, *Evolutionary programming and evolution strategies: Similarities and differences*, in Proc. of 2nd Annual Conf. on Evolutionary Programming, 1993, pp. 11–22.

[3] T. BÄCK AND H.-P. SCHWEFEL, *An overview of evolutionary algorithms for parameter optimization*, Evolutionary Computation, 1 (1993), pp. 1–23.

[4] L. DAVIS, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[5] K. A. DE JONG, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD thesis, University of Michigan, Ann Arbor, 1975.

[6] J. J. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, 1983.

[7] L. J. ESHELMAN AND J. D. SCHAFFER, *Real-coded genetic algorithms and interval schemata*, in Foundations of Genetic Algorithms 2, L. D. Whitley, ed., Morgan-Kauffmann, San Mateo, CA, 1993, pp. 187–202.

[8] D. B. FOGEL, *An introduction to simulated evolutionary optimization*, IEEE Transactions on Neural Networks, 5 (1994), pp. 3–14.

[9] ——, *Evolutionary Computation*, IEEE Press, Piscataway, NJ, 1995.

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

[11] D. E. GOLDBERG, *The theory of virtual alphabets*, in Parallel Problem Solving from Nature, H.-P. Schwefel and R. Männer, eds., New York, 1990, Springer-Verlag, pp. 13–22.

[12] D. E. GOLDBERG AND K. DEB, *A comparative analysis of selection schemes used in genetic algorithms*, in Foundations of Genetic Algorithms, G. J. Rawlins, ed., Morgan-Kauffmann, San Mateo, CA, 1991, pp. 301–315.

[13] W. E. HART, *Adaptive Global Optimization with Local Search*, PhD thesis, University of California, San Diego, May 1994.

[14] J. H. HOLLAND, *Adaptation in Natural and Artificial Systems.*, The University of Michigan Press, 1976.

[15] C. Z. JANIKOW AND Z. MICHALEWICZ, *An experimental comparison of binary and floating point representations in genetic algorithms*, in Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, R. K. Belew and L. B. Booker, eds., San Mateo, CA, 1991, Morgan-Kaufmann, pp. 31–36.

[16] R. JUDSON, M. COLVIN, J. MEZA, A. HUFFER, AND D. GUTIERREZ, *Do intelligent configuration search techniques outperform random search for large molecules?*, International Journal of Quantum Chemistry, (1992), pp. 277–290.

[17] J. MEZA AND M. L. MARTINEZ, *Direct search methods for the molecular conformation problem*, Journal of Computational Chemistry, 15 (1994), pp. 627–632.

[18] H. MÜHLENBEIN, M. SCHOMISCH, AND J. BORN, *The parallel genetic algorithm as function optimizer*, in Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, R. K. Belew and L. B. Booker, eds., San Mateo, CA, 1991, Morgan-Kaufmann, pp. 271–278.

[19] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND W. T. VETTERLING, *Numerical Recipies in C - The Art of Scientific Computing*, Cambridge University Press, 1990.

[20] X. QI AND F. PALMIERI, *Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part I: Basic properties of selection and mutation*, IEEE Trans. on Neural Networks, 5 (1994), pp. 102–119.

[21] ——, *Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part II: Analysis of the diversification role of crossover*, IEEE Trans. on Neural Networks, 5 (1994), pp. 120–129.

[22] G. RAPPL, *Konvergenzraten von Random Search Verfahren zur globalen Optimierung*, PhD thesis, HSBw München, Germany, 1984.

[23] ——, *On linear convergence of a class of random search algorithms*, Zeitschrift f. angew. Math. Mech., 69 (1989), pp. 37–45.

[24] A. RINNOOY KAN, *Probabilistic analysis of algorithms*, Annals of Discrete Mathematics, 31 (1987), pp. 365–384.

[25] G. RUDOLPH, *Convergence analysis of canonical genetic algorithms*, IEEE Transactions on Neural Networks, 5 (1994), pp. 96–101.

[26] N. N. SCHRAUDOLPH AND R. K. BELEW, *Dynamic parameter encoding for genetic algorithms*, Machine Learning, 9 (1992), pp. 9–21.

[27] F. SOLIS AND R.-B. WETS, *Minimization by random search techniques*, Mathematical Operations Research, 6 (1981), pp. 19–30.

[28] T. STARKWEATHER, D. WHITLEY, AND K. MATHIAS, *Optimization using distributed genetic algorithms*, in Parallel Problem Solving from Nature, 1990, pp. 176–185.

[29] V. TORCZON, *Multi-directional search: A direct search algorithm for parallel machines*, Tech. Rep. TR90-7, Rice University, May 1989.

[30] ——, *On the convergence of the multidirectional search algorithm*, SIAM J. Optimization, 1 (1991), pp. 123–145.

[31] ——, *On the convergence of pattern search methods*, Tech. Rep. TR93-10, Rice University, June 1993. Revised Sept. 1994.

[32] A. TÖRN AND A. ŽILINSKAS, *Global Optimization*, vol. 350 of Lecture Notes in Computer Science, Springer-Verlag, 1989.

[33] D. WHITLEY, K. MATHIAS, AND P. FITZHORN, *Delta coding: An iterative search strategy for genetic algorithms*, in Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, R. K. Belew and L. B. Booker, eds., San Mateo, CA, 1991, Morgan-Kaufmann, pp. 77–84.

[34] A. H. WRIGHT, *Genetic algorithms for real parameter optimization*, in Foundations of Genetic Algorithms, G. J. Rawlins, ed., Morgan-Kauffmann, San Mateo, CA, 1991, pp. 205–218.