In Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO-2011) 2011.

Real-Space Evolutionary Annealing

Alan J Lockett
Computer Science Department
University of Texas
Austin, Texas, USA
alockett@cs.utexas.edu

Risto Miikkulainen
Computer Science Department
University of Texas
Austin, Texas, USA
risto@cs.utexas.edu

ABSTRACT

Standard genetic algorithms can discover good fitness regions and later forget them due to their Markovian structure, resulting in suboptimal performance. Real-Space Evolutionary Annealing (REA) hybridizes simulated annealing and genetic algorithms into a provably convergent evolutionary algorithm for Euclidean space that relies on non-Markovian selection. REA selects any previously observed solution from an approximated Boltzmann distribution using a cooling schedule. This method enables REA to escape local optima while retaining information about prior generations. In parallel work, REA has been generalized to arbitrary measure spaces and shown to be asymptotically convergent to the global optima. This paper compares REA experimentally to six popular optimization algorithms, including Differential Evolution, Particle Swarm Optimization, Correlated Matrix Adaptation Evolution Strategies, the real-coded Bayesian Optimization Algorithm, a real-coded genetic algorithm, and simulated annealing. REA converges faster to the global optimum and succeeds more often on two out of three multimodal, non-separable benchmarks and performs strongly on all three. In particular, REA vastly outperforms the real-coded genetic algorithm and simulated annealing, proving that the hybridization is better than either algorithm alone. REA is therefore an interesting and effective algorithm for global optimization of difficult fitness functions.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Parameter learning; I.2.m [**Miscellaneous**]: Genetic algorithms; F.1.2 [**Modes of Computation**]: Probabilistic computation

General Terms

Algorithms

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

Genetic algorithms typically operate in a Markov fashion, with the population for each generation constructed stochastically from the prior population only. As a result, genetic algorithms can discover and then forget high quality regions within the search domain. By forgetting prior generations regardless of quality, the algorithm can fail to exploit crucial information, resulting in suboptimal performance. This problem can be alleviated by selecting individuals for reproduction over the entire pool of previously observed solutions. However, a genetic algorithm with blind non-Markovian selection can become trapped by local optima early on. This issue can be mitigated by combining genetic algorithms and simulated annealing in a novel manner to produce an evoutionary algorithm that solidly outperforms both genetic algorithms and simulated annealing.

Simulated annealing and evolutionary computation have long been regarded as two effective methods of optimization. A number of algorithms have been proposed that hybridize the two methods. One approach utilizes a Metropolis-style acceptance probability to admit inferior solutions into the population in proportion to their quality (see e.g. Das et al. (2007); Mahfoud et al. (1995)). Another approach uses a Boltzmann-like version of proportional selection (Goldberg (1995); Jeong and Lee (1996); Mühlenbein and Mahnig (2002)). However, each of these hybridizations are best regarded as evolutionary algorithms augmented with various aspects simulated annealing.

This paper proposes Real-Space Evolutionary Annealing (REA), an evolutionary algorithm on real vector space that can be alternately viewed as a genetic algorithm with non-Markovian selection or as a method of performing simulated annealing without the Metropolis algorithm. REA exploits a connection between the average effect of proportional selection and the annealed Boltzmann distributions used in simulated annealing. It replaces the operation of a genetic algorithm with a sample from a two-layer mixture model that approximates the Boltzmann distribution. These two layers correspond respectively to selection and variation in a genetic algorithm. Thus REA is both a genetic algorithm and a simulated annealing algorithm.

REA is also related to Estimation of Distribution Algorithms (EDAs), since it builds a global model of the annealing distributions for the fitness function (Pelikan et al. (2002); Mühlenbein et al. (1999)). However, whereas EDAs build models based solely on the best members of the immediately prior generation, REA's models are based on the entire history of observation. Also, while EDAs train graphical

models to recognize a dependency structure within solutions, REA's models are simpler and more direct. As a result, REA may be expected to perform well on fitness functions where the problem structure does not possess obvious intervariable dependencies.

Theoretically, REA converges asymptotically to the true global optima of the fitness function. The proof is given in a parallel paper (Lockett and Miikkulainen (2011)). Experimentally, REA converges fast on a bank of standard benchmarks, including Ackley's function, Whitley's function, and the modified version of Shekel's Foxholes. These benchmarks are multi-modal and non-separable and are therefore difficult for many optimization methods. However, based on its efficient sampling, REA performs quite well on them. In a comparison with Differential Evolution (DE), Particle Swarm Optimization (PSO), Correlated Matrix Adaptation Evolution Strategies (CMA-ES), the real-coded Bayesian Optimization Algorithm (rBOA), a real-coded genetic algorithm (GA), and simulated annealing (SA), REA outperforms its competitors on two of these three problems and exhibits strong performance on all three. In particular, REA vastly outperforms the real-coded genetic algorithm and simulated annealing, proving that the hybridization is better than either algorithm alone. REA is therefore an interesting and effective algorithm for global optimization of difficult fitness functions.

2. BACKGROUND

Before describing REA and the experiments in detail, some motivating background in simulated annealing and its connection to proportional selection will be reviewed.

2.1 Simulated Annealing

Simulated annealing is a general optimization algorithm that employs properties of statistical mechanics to locate minima of a given fitness function (Kirkpatrick et al. (1983); Bertsimas and Tsitsiklis (1993)). The usual analogy is that of crafting a metallic artifact by repeatedly shaping it at different temperatures. At high temperatures, the metal is malleable and easy to shape, but does not readily remain in detailed configurations. As the temperature is gradually lowered, more refined and delicate shapes become possible, but the overall shape is increasingly fixed.

At the core of the simulated annealing algorithm is the Boltzmann distribution. At time n, simulated annealing samples from a distribution given by

$$\mathcal{A}_n^f(dx) = \frac{1}{Z_n} \exp\left(-\frac{f(x)}{T_n}\right) dx, \tag{1}$$

where f is the fitness function, Z_n is a normalizing factor known as the *partition function*, and T_n is a sequence of temperatures with $T_n \to 0$. The sequence T_n is known as the *cooling schedule*. The distribution $\mathcal{A}_n^f$ will be referred to as an *annealing distribution* in this paper. Simulated annealing samples from $\mathcal{A}_n^f$ repeatedly using the Metropolis algorithm (Metropolis et al. (1953); Hastings (1970)). The process begins with a proposed solution x. At each time step, a proposal distribution $\mathbb{Q}$ is used to sample x_n. The proposed solution x is replaced with x_n with probability $\exp\left(-\max\left\{0, f(x) - f(x_n)\right\}/T_n\right)$. For each fixed temperature T_n the algorithm will converge to a sample from $\mathcal{A}_n^f$. As $n \to \infty$, $\mathcal{A}_n^f$ converges in probability to a distribution that samples directly from the optimal points of f (Kirkpatrick et al. (1983)).

Simulated annealing can be shown to converge asymptotically to the global optima of the fitness function, subject to conditions on the cooling schedule (Hajek (1988)). For combinatorial problems, Hajek (1988) showed that simulated annealing converges if the cooling schedule is set according to $T_n \propto 1/\log n$. In practice, simulated annealing has been used effectively in several science and engineering problems. However, its sensitivity to the proposal distribution and the cooling schedule means that it is not a good fit for all optimization problems.

Surprisingly, traditional genetic algorithms are connected with simulated annealing through an analysis of the average performance of proportional selection. This connection is exposed by trivial manipulations of a previous result of Mühlenbein and Mahnig (2002); the details are discussed next.

2.2 Expected Proportional Selection

One version of selection in genetic algorithms is *proportional selection*, where individuals in the prior population are selected proportionally to their observed fitness. Unexpectedly, there is a deep connection between simulated annealing and proportional selection in standard genetic algorithms. Much like simulated annealing, proportional selection sharpens the fitness function implicitly with each generation, so that on averaging over population trajectories the selection operator asymptotically places probability one on the optima of the fitness function. The following argument for discrete spaces is derived from Mühlenbein and Mahnig (2002) in the context of maximizing a fitness function; analogues to this result hold in arbitrary measure spaces.

Proportional selection at the n^{th} time step is given by $\mathcal{S}_f^n(x) \propto f(x) N_x^{n-1}$, where $\mathcal{S}_f^n(x)$ is the probability of selecting x at time n, and N_x^n is a random variable indicating the number of copies of the solution x in the population at time n. Taking the expected value over N_x^n,

$$\mathbb{E}\left[\mathcal{S}_f^n(x)\right] \propto f(x) \mathbb{E}\left[N_x^{n-1}\right]. \tag{2}$$

The expected value on the left is also a probability distribution over x and therefore a selection rule, here termed *expected proportional selection*. It is possible to imagine an evolutionary algorithm where each successive population is sampled from just this rule. This algorithm is a one-stage, selection-only genetic algorithm; because expected proportional selection averages over all individuals, no variation is required.

In such an algorithm, if the initial population is selected uniformly at random (assuming this is possible), then $\mathbb{E}\left[N_x^0\right]$ is a constant, so

$$\mathbb{E}\left[\mathcal{S}_f^1(x)\right] \propto f(x). \tag{3}$$

By definition, $\mathbb{E}[\mathcal{S}_f^n(x)] = \mathbb{E}[N_x^n]/K$ where K is the population size, since N_x^n/K is just the proportion of the population taking the value x. Applying this fact to the recursion in Equation 2 yields $\mathbb{E}[S_f^n(x)] \propto f(x)^n$. Thus expected proportional selection sharpens the fitness function. Introducing $g(x) \equiv -\log\left(f(x)\right)$,

$$\begin{aligned} \mathbb{E}\left[S_f^n(x)\right] &\propto \exp\left(-g(x)\right)^n \\ &= \exp\left(-\frac{1}{n^{-1}} g(x)\right) \end{aligned} \tag{4}$$

Comparing Equation 1 to Equation 4, expected proportional selection is found to have an annealing distribution on $-\log f$ with cooling schedule $T_n = n^{-1}$. Since the logarithm is monotonic, the maxima of f are the minima of g.

Expected proportional selection is not a feasible selection rule because it requires total knowledge of the fitness function a priori. If such knowledge were possible, there would be no need for evolutionary computation; the optima would already be known. Expected proportional selection could be estimated by averaging over the trajectories of several different runs of a genetic algorithm, but the number of trajectories required for a good estimate would be intractably large.

Instead, genetic algorithms approximate expected proportional selection with proportional selection. Taken on its own, proportional selection has quite different properties from expected proportional selection. Proportional selection can only select the small finite set of solutions contained in the prior population. Right from the beginning and throughout, expected proportional selection can select any point with a positive fitness value. Expected proportional selection is not computable in general, however. There is simply no way to get a meaningful estimate of $\mathbb{E}[N_x^n]$ except in the most trivial of problems.

REA exploits this theoretical relationship between simulated annealing and genetic algorithms to create a hybridized algorithm that merges qualities of both algorithms by selecting solutions according to an approximation of the annealing distributions and then mutating the selected individual through standard evolutionary techniques.

3. REAL EVOLUTIONARY ANNEALING

REA is two-stage genetic algorithm for Euclidean space with selection and variation phases. In REA, the selection phase approximates a Boltzmann distribution with increasing accuracy. Let f be a fitness function to be minimized, and let S_n^k represent the k^{th} selected individual at the n^{th} time step. The sequence of populations generated by REA will be represented by $P_n = \left(P_n^k\right)$, so that P_m^k is the k^{th} member of the population from the m^{th} generation. Then the selection probability for REA is given by

$$\mathbb{P}\left(S_n^k = x\right) = \begin{cases} 0 & x \notin \bigcup_{m<n,k} P_m^k \\ \frac{\xi_n}{c_n(x)} \exp\left(-\frac{1}{T_n} f(x)\right) & \text{otherwise,} \end{cases} \tag{5}$$

where ξ_n is a normalizing factor, and T_n is a cooling schedule with $T_n > 0$ and $T_n \to 0$. The function $c_n(x)$ adjusts the selection probabilities to avoid biasing the distribution towards previously sampled points as discussed below. For the experiments below, the cooling schedule is given by $T_n^{-1} = \eta \log n$, where η is a problem specific learning rate and $\log n$ is a traditional cooling rate for simulated annealing [Hajek (1988)].

It is a distinguishing feature of REA that every member of every previous population is assigned positive probability by Equation 5. That is, selection in REA is non-Markovian; any previously observed solution from any generation can be selected. It is this fact that makes REA different from other evolutionary algorithms and other hybrids of simulated annealing and genetic algorithms.

Once S_n^k is selected, the variation phase of REA perturbs the selected individual according to a mutation distribution.

Algorithm 1 Algorithm for Real Evolutionary Annealing

```
N, the number of generations
K, population size
Sample P_1^k from an initial distribution
for n ← 2 to N do
  T_n ← 1/(η log n)
  σ_n ← σ exp(−n^α + sin x)
  total ← 0
  for m ← 1 to n − 1 do
    for k ← 1 to K do
      compute c_n(P_m^k)
      y_m^k ← c_n(P_m^k)^{-1} exp(−(1/T_n) f(P_m^k))
      total ← total + y_m^k
      z_m^k ← total
    end for
  end for
  for k ← 1 to K do
    r ← Uniform(0,1)
    w, j ← min{m, k : z_m^k < r × total}
    S_n^k ← P_w^j
    P_n^k ← N(S_n^k, σ_n^2)
  end for
end for
```

Gaussians are used in this paper, but in principle different variation distributions could be used depending on the problem and the space. With selection and variation, the population process for REA is distributed according to

$$P_n^k \sim \mathcal{N}\left(S_n^k, \sigma_n^2 I\right). \tag{6}$$

The variation σ_n^2 should begin at a high level during initial generations in order to quickly explore the space and should then decay in later generations in order to refine the solution. For the experiments below, static decay rates for σ_n were used: $\sigma_n = \sigma \exp(-n^\alpha + \sin n)$ with σ and α as parameters, discussed below. The sine term causes the variation to oscillate between relatively high and low values within the context of overall decay; this feature helps the algorithm to escape local minima and plateaus by attempting several substantially different mutations in succession.

The function $c_n(x)$ in the denominator of Equation 5 exists to prevent a particular type of premature convergence that would otherwise occur in REA. Suppose $c_n(x) = 1$. Consider the situation in which REA discovers a new local minimum as the current maximum score. Selection will favor exploitation of this new minimum, and within a few generations the surrounding region will be filled with a large number of test points looking for a better solutions close to the current local minimum. Since many of these test points will have comparable fitness, they will further bias selection to favor the same region. In a short period of time, the probability of REA selecting a point outside of that region would become small. To avoid this situation, the function $c_n(x)$ is set to count the number of points close to x. In this paper, $c_n(x)$ counts the number of previously observed points inside of a hypercube of side length $2\sigma_n$ centered at x, with x included in the count. The value of $c_n\left(P_m^k\right)$ can be efficiently computed in $O(\log nK)$ time using a database index. A hypercube was chosen over a hypersphere because

hypersphere computations would necessarily require linear time.

The REA algorithm is given explicitly in Algorithm 1. The algorithm simply formalizes the process of repeatedly sampling S_n^k and P_n^k according to the formulae above. The complexity of the algorithm is $O\left(N^2K^2 \log NK\right)$ where N is the number of generations, K is the population size, with NK fitness evaluations. The sample points P_n^k can be stored in a database index database, so that the cost of sampling a new population is $O\left(K \log NK\right)$. The most expensive portion of the computation is the recomputation of the mixing probabilities that must be performed for each generation. At low temperatures, as $n \to \infty$, this computation can be made more efficient by pruning points that are unlikely to be selected. It is important to recognize that while REA introduces substantial overhead relative to other evolutionary methods, this overhead is mostly independent of the fitness function. For complex problems, where fitness function evaluation can take minutes rather than milliseconds, the overhead for REA is quite acceptable, adding perhaps 5–10% to the overall running time versus simpler algorithms.

The three learning parameters for REA as described in this paper are η, α, and σ. The learning rate η controls the sensitivity of selection to the fitness value by scaling the fitness prior to computing the mixture distribution. If η is large (> 1), selection will be more sensitive to smaller fluctuations in the fitness. If η is small (< 1), selection will be less sensitive to large fluctuations in the fitness. A value of $\eta = 1$ is a good default for this parameter; the optimal value will depend on the volatility of the fitness function around the optima.

The parameters α and σ control the variation. A large value for σ will cause exploration to essentially ignore selection in early generations, promoting exploration at the expense of refinement. A smaller value for σ will tend to limit the results to the quality of the initial population. The value for σ should be chosen to balance these effects on the particular problem and domain. The value for α controls the rate of decay in the variance. A small value for α favors prolonged exploration but inhibits refinement. A high value for α will cause the algorithm to quickly focus on a small region around the selected points. Based on the experiments, a good rule of thumb is to set the standard deviation σ to be equal to about half the width of the space and to set α between 1/4 and 1/3.

Provided that $\sigma_n \to 0$, the sample distribution for P_n^k converges to the global optima of f in probability as $n \to \infty$. In brief, as the space is completely explored by REA, the population process eventually places a point on each of the rationals, and the integrability of f then implies convergence with $\mathcal{A}_n^f$ for each fixed n as the variance decays. Since $\mathcal{A}_n^f$ converges to the optima of f in probability, so does REA. The actual proof of convergence can be found in parallel work [Lockett and Miikkulainen (2011)].

In general, the population process for REA has a mixture distribution at each generation. Letting $P^n \equiv \bigcup_{m<n,k} P_m^k$ be the set of previously observed individuals and $p_n(x) \equiv \mathbb{P}\left(S_n^k = x\right)$, the distribution for the individual P_n^k is given by

$$\mathbb{P}_n(dx) = \sum_{a \in P^n} p_n(a)\, \nu_n^a(dx), \tag{7}$$

where ν_n^a is the variation distribution, shifted so that it is centered at a. In this paper, $\nu_n^a = \mathcal{N}\left(a, \sigma_n^2 I\right)$. A more general implementation would allow arbitrary distributions to be used for different problems. For example, recombination could also be built into the variation distribution if the optimization problem exhibits symmetries that would be exploited by recombination. Note that alternative versions are not limited to Euclidean space. REA can be applied to any measure space admitting a finite measure, including arbitrary length bit strings, neural networks, and Bayesian networks. An arbitrary implementation in this fashion is discussed in parallel work (Lockett and Miikkulainen (2011)).

A sequence of mixture distributions that adds a mixing point for each new sample is known as a mixture sieve. The convergence properties of mixture sieves have been studied for approximation of smooth densities by Genovese and Wasserman (2000) and by Ghosal and van der Vaart (2001). When the mixture sieve is determined by a maximum likelihood estimate, an upper bound on the convergence rate to the target distribution is given by $C\left(\frac{\log n}{n}\right)^{\frac{1}{4}}$. This result does not imply anything definite about the convergence rate of REA, but it does provide reason to expect that REA may converge quickly.

Because REA does not assume that the problem is highly structured, it is expected that it will perform well on fitness functions that possess little global structure or symmetry. For this purpose, REA was tested on three chaotic, multimodal, non-separable fitness function common in the literature and compared to other evolutionary algorithms. The benchmarks and comparisons are presented in the next section.

4. EXPERIMENTAL SETUP

REA was tested on three standard benchmarks: Ackley's function, Whitley's function, and a modified version of Shekel's Foxholes (see e.g. Ali et al. (2005)).

Ackley's:

$$\sum_{i=1}^{d-1} e^{-0.2}\sqrt{x_i^2 + x_{i+1}^2} + 3\cos(2x_i) + 3\sin(2x_{i+1}) \tag{8}$$

Domain: $x_i \in (-5.12, 5.12)$
Minimum: -13.37957500565419

Whitley's:

$$\sum_{i=1}^{d}\sum_{j=1}^{d} \frac{w(x_i, x_j)^2}{4000} - \cos(w(x_i, x_j)) + 1, \tag{9}$$

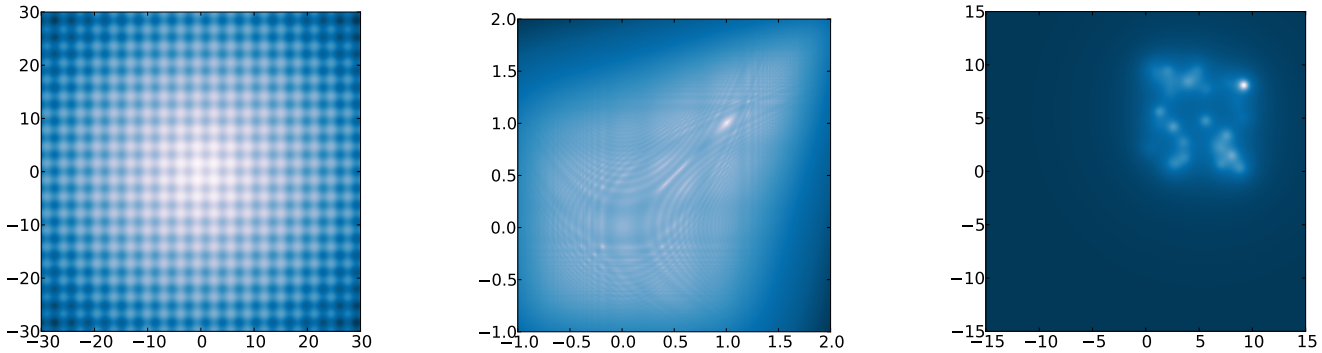
$$\text{with } w(y,z) = 100\left(y^2 - z\right)^2 + (1-z)^2$$

Domain: $x_i \in (-30, 30)$
Minimum: 0

Modified Shekel's Foxholes:

$$\sum_{i=1}^{30} \frac{1}{\sum_{j=1}^{d}(x_j - a_{ij})^2 - c_i} \tag{10}$$

Domain: $x_i \in (-15, 15)$
Minimum: -10.40561723899245



(a) Ackley's (b) Whitley's (c) Shekel's Foxholes

Figure 1: Heat maps for the three benchmark functions in two dimensions ($d = 2$). Each heat map is scaled to show the critical region. Lighter colors indicate lower and therefore more optimal values. Whitley's is scaled to show the critical region; the function rises sharply outside of this region. These three functions are multimodal and non-separable and are quite difficult for most optimization methods.

For Shekel's foxholes, the matrices $A = \{a_{ij}\}$ and the vector $c = \{c_i\}$ can be found in Ali et al. (2005). Figure 1 shows heat maps for each function in two dimensions to graphically display the properties of the benchmark; lighter colors indicate lower and therefore more optimal values. For Whitley's, the heat map was scaled to show detail in the critical region. The experiments were performed in five dimensions ($d = 5$) on a search space whose domain was the hypercube with components as given above.

In order to establish the performance of REA relative to other evolutionary algorithms, experiments were run with six other algorithms: (1) simulated annealing (SA), (2) a genetic algorithm (GA), (3) an evolution strategy (CMA-ES), (4) differential evolution (DE), (5) particle swarm optimization (PSO), and (6) the real-coded Bayesian Optimization Algorithm (rBOA). The algorithms selected for comparison cover a broad spectrum of evolutionary algorithms and represent a general sampling of the current state of the art. These algorithms are known to be effective on a wide array of fitness functions and most of them perform reasonably well on the selected benchmarks.

Simulated annealing was performed on a single chain for $25,000$ function evaluations, randomly restarting with probability 0.001 at each step. The cooling schedule was linear. All algorithms except simulated annealing used a population size of 100 and were trained for 250 generations. The GA used linear ranking selection with selection pressure at 1.8. Crossover was uniform, and mutation was a Gaussian with variance 0.05 (Wright (1991); Baker (1985); Syswerda (1989)). The ES was a $(10 + 90)$-ES using Correlated Matrix Adaptation (CMA) to evolve the mutation parameters (Hansen and Ostermeier (1996)). DE was trained with a crossover probability of 0.9 for Whitley's and 0.2 for Ackley's and Shekel's and with a learning rate of 0.5 for Ackley's and 0.9 for Whitley's and Shekel's (Storn and Price (1995)). PSO was trained with both the global and local adaptation rates set to 0.25 (Eberhart and Kennedy (1995)). rBOA was implemented as described in Ahn et al. (2006). The parameters for REA were set with $\eta = 1$ for Ackley's and Shekel's Foxholes, and $\eta = 0.1$ for Whitley's. The initial standard variance σ was set to 32 for Ackley's, 8 for Whitley's, and 16 for Shekel's Foxholes. The variance decay was set at $\alpha = 1/3$ for all problems. For all of the algorithms, parameters were set according to the literature where available and hand-tuned otherwise to optimize performance.

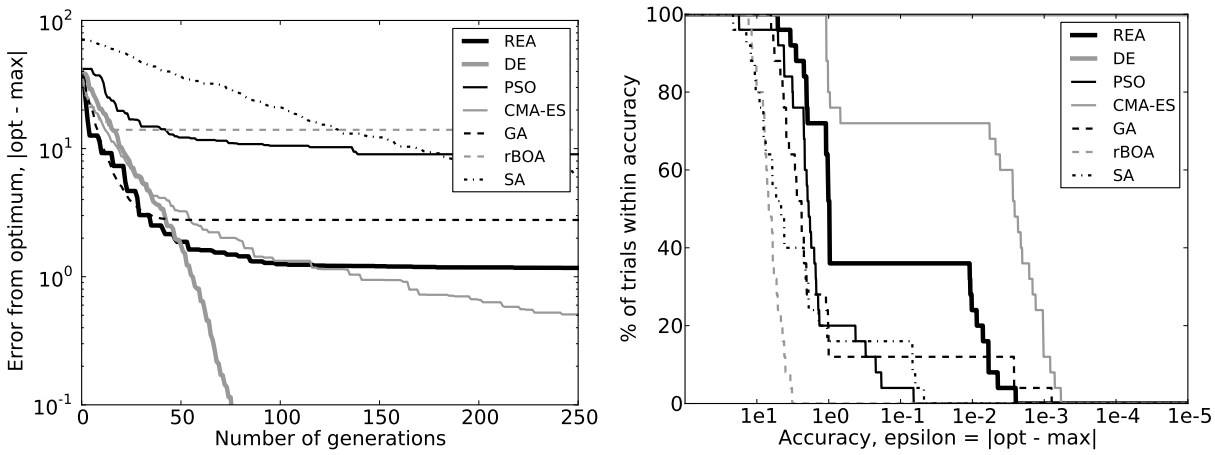
For REA, the variance was set quite high at the outset, with a standard deviation nearly as large as the space. This large variance promotes extensive exploration of the space in early generations. The value $\alpha = \frac{1}{3}$ causes the variance to decay quickly. The leaning rate η performed well generally with $\eta = 1$, but for Whitley's function, REA required a lower learning rate so that probability mass was equitably distributed between competing local minima at a critical stage. All algorithms were run on all benchmarks 25 times.

5. EXPERIMENTAL RESULTS

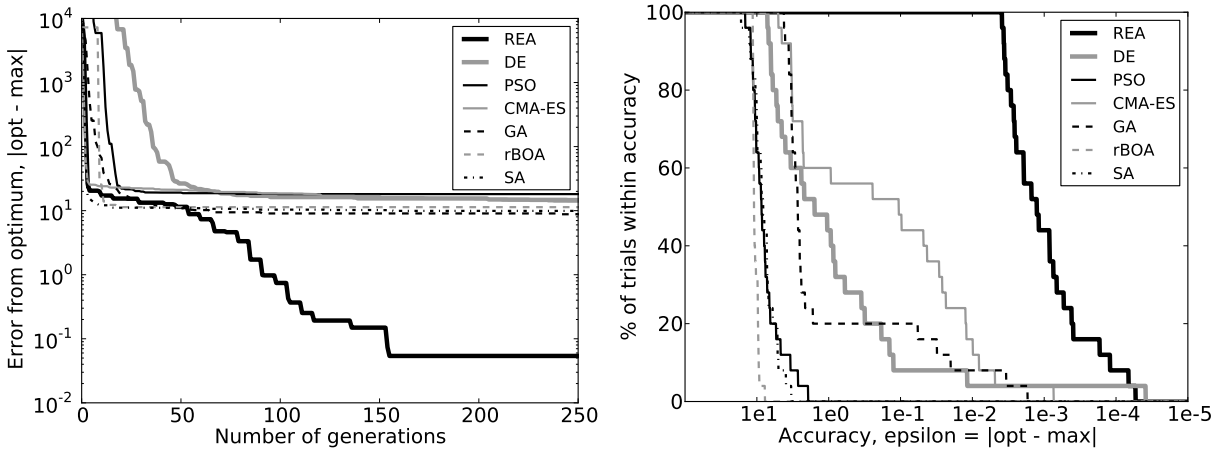
REA did indeed perform very well on these benchmarks. The left-hand column of Figure 2 displays average learning curves for each algorithm on a semilog scale. These graphs thus display both the degree and the rate of convergence to the optimum. On Whitley's function and Shekel's Foxholes, REA outperforms all competitors both in terms of the quality of optima and speed of convergence. On Ackley's function, REA still displays faster convergence in the first 50 generations, but is outperformed by DE and later by CMA-ES as well. The reasons for this will be explored below.

The right-hand column of Figure 2 shows the percentage of trials in which each algorithm reached various levels of accuracy within 250 generations, shown in log scale with higher accuracy on the right side of the x-axis. On Whitley's function, REA achieves high accuracy consistently on all trials. On Shekel's Foxholes, REA is able to locate the true optimum with higher probability than any of the other algorithms. On suboptimal runs, its performance is comparable to DE and CMA-ES. For Ackley's function, REA performs well in general, but becomes trapped in a strong local optimum about 50% of the time.

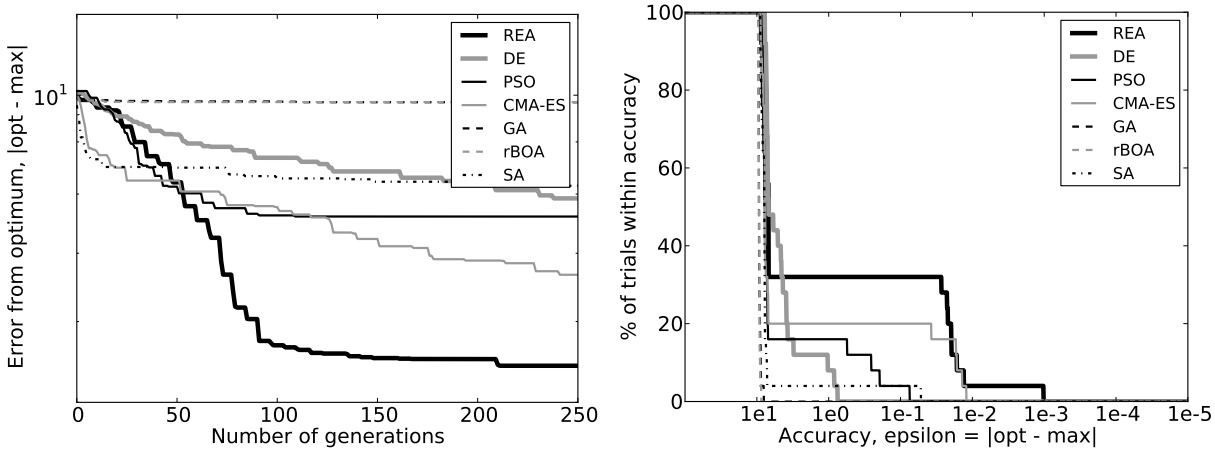
A numerical summary of the results is shown in Table 1. This table shows the number of the 25 trials in which each algorithm reached an accuracy threshold of $\epsilon < 0.02$.



(a) Ackley's



(b) Whitley's



(c) Shekel's Foxholes

Figure 2: Error rates for REA and five other algorithms on Ackley's function, Whitley's function, and Shekel's Foxholes. The left panel shows the average learning curve for each algorithm on the benchmarks, with the number of generations on the x-axis and the average distance to the global optimum at each generation on the y-axis in log scale. These graphs were generated by averaging the global best of each of the 25 trials after each generation; lower values are better. The right panel shows the percentage of trials coming within various distances from the optimum. The x-axis shows various error rates in a sign-reversed log scale, so that the values to the right indicate exponentially smaller error. The y-axis shows the percentage of trials achieving each level of accuracy. For the right panel, higher values are better. REA has the best performance on Whitley's and Shekel's Foxholes and performs strongly on Ackley's as well. REA is characterized by faster convergence than the competing algorithms in general.

Table 1: Number of Successful Trials ($\epsilon < 0.02$)

	Ackley's	Whitley's	Shekel's
REA	9	25	7
DE	25	2	0
CMA-ES	18	6	4
PSO	0	0	0
GA	3	2	0
rBOA	0	0	0
SA	0	0	0

Of the three benchmarks used in this research, Ackley's function possesses the most regular structure. As evident from Figure 1(a), Ackley's is generally spherical and has local optima evenly spaced along axis-parallel lines. There are two reasons why DE performs extremely well in this environment. First, DE proposes changes to its current population that move along axis-parallel lines. Second, because the local optima for Ackley's are spread evenly around the global optima, chances are high that change proposed by DE will point from a local optima towards the global optimum. Thus Ackley's is shaped in a manner that favors DE, accounting for the strong performance of DE on this benchmark.

REA locates good local optima very quickly, initially at a rate even faster than DE. But as the variance narrows, it becomes more difficult for REA to skip past the final local optima to reach the global optimum, and it becomes entrapped on approximately half of the trials. CMA-ES, by contrast, can expand its variance to escape local optima and is therefore able to catch up to and ultimately surpass REA.

In contrast, REA performed most strongly on Whitley's function. The surface of this function is quite irregular, but the variation among these irregularities is small in magnitude relative to the search domain; outside of the region shown in Figure 1(b), Whitley's function increases exponentially. This surface structure was well-suited to the cooling schedule used in REA. Because of the magnitude of function values, it was possible to set the learning rate η to 0.1, smoothing the function at the origin. On some other problems (such as Shekel's), lowering the learning rate causes REA to stagnate because it is unlikely to select the current best solutions. But on Whitley's, the disparity of function values resulted in REA selecting the best members early on, so that the critical region of the fitness function was quickly discovered. Then, the lower learning rate forced more thorough exploration of the local optima, so that REA was invariably able to locate the true optima within accuracy $\epsilon < 0.005$.

Shekel's Foxholes is the most difficult of the three benchmarks used in this paper. As the heat map in Figure 1(c) shows, the local optima are quite sparse and somewhat distant from each other. There is effectively no systematic means to approach the global optimum; it must be discovered by chance. Yet the region in which the global optimum is located is narrow and difficult to find with few function evaluations. None of the algorithms tested were successful even half of the time with the alotted $25,000$ function evaluations. On this benchmark, REA performed the best, locating the global optimum within accuracy $\epsilon < 0.02$ in about a third of the trials, making it twice as likely to find the global optimum as CMA-ES, the next best performer, which came within $\epsilon < 0.02$ in four of the 25 trials. None of the other algorithms reached this level of accuracy, though PSO and DE each came within $\epsilon < 1$ in four and three trials, respectively. Since Shekel's Foxholes contains the least identifiable problem structure of the three benchmarks tested, REA's performance on this benchmark validates the use of REA for difficult problem domains.

6. DISCUSSION AND FUTURE WORK

This paper has shown that REA is an effective approach to difficult benchmarks. It compares favorably with other evolutionary algorithms, converging quickly and reliably on challenging problems. Notably, REA drastically outperforms a genetic algorithm and simulated annealing, the two algorithms that inspired it.

Because REA is a new method, experiments still need to be performed to test the effect of varying certain features of the basic algorithm. For instance, REA uses an oscillating variance decay. The oscillation appears to promote better exploration of the search space. However, this conjecture needs to be verified experimentally. Also, parallel work suggests that the counting approach for setting $c_n(a)$ might improved upon by using a series of successive partitions of the search space to set the value of $c_n(a)$ to the area of the region containing the point a (Lockett and Miikkulainen (2011)). While training on Ackley's, REA has a blind spot on the region around the global optimum; REA is unaware that this region has not been well explored. Partitions could also be used to dynamically scale the variance at each selected point based on the degree to which the surrounding region has been explored, potentially allowing for the discovery of the true optimum.

Partitioning is also more computationally efficient than the counting approach; where as the counting approach requires $O\left(N^2 \log N\right)$ time, partitioning requires at most $O\left(N^2\right)$ time. Also, partitioning builds a binary tree over the space, and it may be possible to sample on this structure, further reducing the time requirement to $O\left(N \log N\right)$, assuming the tree can be balanced.

Finally, REA may perform better with Cauchy rather than Gaussian variance, since the fatter tails of the Cauchy distribution may allow it to avoid local optima. All of these features need to be tested and compared with each other to determine how they affect the basic algorithm.

The annealing selection mechanism in REA was based on an analysis of expected proportional selection. Proportional selection was developed earlier in the history of genetic algorithms. It has since been superseded by other selection mechanisms such as tournament selection and ranking selection. When a population is close to a local optimum, the ratio of fitnesses among the population becomes close to one. Because of this, proportional selection often fails to select the best known solution, and thus has difficulty refining the fitness value near local optima. Tournament and ranking selection overcome this difficulty by selecting solutions according to their rank in the population rather than their raw fitness. Being based on proportional selection, REA suffers from the same difficulty when refining the final solution. REA could be adapted to use an annealed version of tournament selection by storing the candidate values in a sorted, balanced binary tree. It may then be possible to derive a tree-sampling algorithm that samples tournament selection

with a selection pressure that grows as the temperature decreases.

Further, REA can be generalized to arbitrary measure spaces. Since the Boltzmann-style selection employed by REA depends solely on the fitness values, only the variation mechanism needs to be changed. A REA-like algorithm could be developed for training neural networks or game strategies. These generalizations would enable REA to be applied to more complex problem domains.

On the theoretical side, while it has been proven that REA converges asymptotically, there is still a need to understand the convergence rate better. It is clear from Figure 2 that convergence rates in REA are comparable to or better than other popular evolutionary methods on certain problems. These results need to be accounted for theoretically. Such an account may make it possible to identify the problem domains in which REA can be expected to perform best.

7. CONCLUSION

This paper proposes REA as a provably convergent evolutionary algorithm that can be viewed as a method of performing simulated annealing or as a genetic algorithm with non-Markovian selection. In experiments with three standard optimization benchmarks, REA compares favorably with existing evolutionary methods. Additionally, REA outperforms both genetic algorithms and simulated annealing on the benchmarks by a wide margin. REA therefore presents a promising new approach to evolutionary computation on difficult and unstructured domains.

References

Ahn, C., Ramakrishna, R., and Goldberg, D. 2006. Real-coded bayesian optimization algorithm. In *Towards a New Evolutionary Computation*, J. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds. Studies in Fuzziness and Soft Computing, vol. 192. Springer Berlin / Heidelberg, 51–73.

Ali, M. M., Khompatraporn, C., and Zabinsky, Z. B. 2005. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization 31*, 635–672. 10.1007/s10898-004-9972-2.

Baker, J. E. 1985. Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their Applications*.

Bertsimas, D. and Tsitsiklis, J. 1993. Simulated annealing. *Statistical Science 8,* 1.

Das, S., Konar, A., and Chakraborty, U. K. 2007. Annealed differential evolution. In *IEEE Conference on Evolutionary Compution (CEC-2007)*.

Eberhart, R. C. and Kennedy, J. 1995. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*. Nagoya, Japan.

Genovese, C. and Wasserman, L. 2000. Rates of convergence for the gaussian mixture sieve. *Annals of Statistics 28,* 4.

Ghosal, S. and van der Vaart, A. W. 2001. Entropies and rates of convergence for maximum likelihood and bayes estimation for mixtures of normal densities. *Annals of Statistics 29,* 5.

Goldberg, D. E. 1995. A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems 4*.

Hajek, B. 1988. Cooling schedules for optimal annealing. *Mathematics of Operation Research 13,* 4.

Hansen, N. and Ostermeier, A. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. Morgan Kaufmann, 312–317.

Hastings, W. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika 57,* 1.

Jeong, I. and Lee, J. 1996. Adaptive simulated annealing genetic algorithm for system identification. *Engineering Applications of Artificial Intelligence 9,* 5, 523 – 532.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. 1983. Optimization by simulated annealing. *Science 220,* 4598.

Lockett, A. and Miikkulainen, R. 2011. Measure-theoretic evolutionary annealing. In *Proceeedings of the 2011 IEEE Congress on Evolutionary Computation (CEC-2011)*.

Mahfoud, S., Mahfoud, S. W., Goldberg, D. E., and Goldberg, D. E. 1995. Parallel recombinative simulated annealing: A genetic algorithm.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. 1953. Equations of state calculations by fast computing machines. *Journal of Chemical Physics 21,* 6.

Mühlenbein, H. and Mahnig, T. 2002. Mathematical analysis of evolutionary algorithms. In *Essays and Surveys in Metaheuristics, Operations Research/Computer Science Interface Series*. Kluwer Academic Publisher, 525–556.

Mühlenbein, H., Mahnig, T., and Rodriguez, A. O. 1999. Schemata, distributions, and graphical models in evolutionary optimization. *Journal of Heuristics 5*.

Pelikan, M., Goldberg, D., and Lobo, F. 2002. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications 21*.

Storn, R. and Price, K. 1995. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces.

Syswerda, G. 1989. Uniform crossover in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*.

Wright, A. H. 1991. Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms*. Morgan Kaufmann, 205–218.