Do Multiple Trials Help Univariate Methods?

Daniel Rothman
Dept. Computer Science
George Mason University
4400 University Dr., MSN 4A5
Fairfax, VA 22030
drothma1@gmu.edu

Sean Luke
Dept. Computer Science
George Mason University
4400 University Dr., MSN 4A5
Fairfax, VA 22030
sean@cs.gmu.edu

Keith Sullivan
Dept. Computer Science
George Mason University
4400 University Dr., MSN 4A5
Fairfax, VA 22030
ksulliv2@cs.gmu.edu

***Abstract*—Cooperative Coevolutionary Algorithms (CCEAs) and Univariate Estimation of Distribution Algorithms (Univariate EDAs) are closely related algorithms in that both update marginal distributions/populations, and test samples of those distributions/populations by grouping them with collaborators drawn from elsewhere to form a complete solution. Thus the quality of these samples is context-sensitive and the algorithms assume low linkage among their variables. This results in well-known difficulties with these methods. While EDAs have commonly overcome these difficulties by examining multivariate linkage, CCEAs have instead examined basing the fitness of each marginal sample on the maximum of several trials. In this study we examine whether multiple-trial CCEA approach is really effective for difficult problems and large numbers of subpopulations; and whether this approach can be used to improve Univariate EDAs as well.**

I. Introduction

In this paper we explore the effects of multiple trials on the effectiveness of *univariate* or *marginal* evolutionary algorithms. By this we mean those algorithms which do not search for single candidate solution but rather search, in parallel, for separate parts of the solution, and test those parts by combining them with one another. The two most common examples of this are Cooperative Coevolutionary Algorithms (CCEAs) and Univariate (or Marginal) Estimation of Distribution Algorithms (EDAs).

Both CCEAs and Univariate EDAs are designed for challenging, high dimensional problems for which linkage among variables is presumed to be low. Their approaches are of course very different. CCEAs, like traditional EAs, develop populations of samples, one for each subsolution, while Univariate EDAs maintain marginal distributions which estimate these populations. However the two methods are very similar in how they update these "populations": by taking samples from each of the populations (or distributions) and testing them together with samples from other populations (or distributions) to form complete solutions.

This similarity results in similar game-theoretic pathologies: both CCEAs and Univariate EDAs tend to converge to suboptimal solutions because each population, or distribution, can only see its own projection of the problem. In the Univariate EDA realm the solution to this problem has generally been to move to *multivariate* EDAs which consider joint distributions among variables. CCEAs allow more general solution representations and so cannot really do this: instead CCEA research has tended to set the fitness of a sample to the maximum over some N trials rather than a single trial. At the limit this has been shown to guarantee reduction of the CCEA to something more approximating an EA [1].

In other work [2] we have examined the performance of various CCEA parameters for hard problems. Here we continue this work by examining the effect of multiple trials on CCEAs and on Univariate EDAs to see if this effect is significant for CCEAs and if it might be transferred to the Univariate EDA case as well. To date EDAs have rarely made use of mechanisms employing multiple trials. Research in EDAs has been focused on explicit modeling of multivariate joint distributions. In this paper we also explore the idea of applying multiple trials to existing univariate EDAs as an alternative to multivariate EDAs.

Our approach to computing fitness assessment from multiple trials is a procedure we call *shuffling*, which proved easily the best approach in our previous CCEA work for nontrivial/non-separable functions. In CCEAs this approach is essentially the selection of collaborators *without* replacement: each subpopulation is shuffled randomly, then collaborators from each subpopulation are paired off. In a Univariate EDA the situation is similar: we sample M times from each distribution, then treat these samples essentially as "subpopulations" in the CCEA shuffling case. The details of the methodology are discussed in Sections IV for CCEAs, and V for Univariate EDAs.

We test these methods using a variety of target functions, shown in Table I. We chose hard, high dimensional, real-valued target functions to provide a challenging testbed. The CCEA methods were tested with a subset of these methods, consistent with previous work. We then expanded on them to investigate the Univariate EDA situation more widely.

In the paper we begin with related work and description of these target functions, then move on to CCEA experiments, then finally to the Univariate EDA experiments.

II. Related Work

Estimation of Distribution Algorithms (EDAs) replace the traditional evolutionary computation population with a statistical distribution of an infinite population. At each generation, individuals are generated from this distribution, and their fitnesses are used to update the distribution such that high

Problem	*Description*	*Bounds*
Rosenbrock*	$\sum_{i=1}^{n-1}(1-x_i)^2 + 100(x_{i+1}-x_i^2)^2$	$-2.048 \leq x_i \leq 2.048$
Rastrigin*	$10n\sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i)$	$-5.12 \leq x_i \leq 5.12$
Schwefel*	$\sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$-512.03 \leq x_i \leq 511.97$
Weierstrass	$10\left(\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{11}\frac{1}{2^k}\cos(2\pi 3^k(z_i+1/2)) - f_0\right)^3 + f_{pen}(\vec{x}) + f_{opt}$	$-5 \leq x_i \leq 5$
	$z_i = \mathbf{R_1}\mathbf{\Lambda}^{1/100}\mathbf{R_2}\mathbf{R_1}(T_{osz}(\vec{x}-\vec{x}^{opt}))$	
Gallagher's Gaussian 101-me Peaks	$T_{osz}\left(10 - \max_{i=1}^{101} w_i e^{-\frac{1}{2n}(\vec{x}-\vec{y}^{(i)})^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{C}_i\mathbf{R}(\vec{x}-\vec{y}^{(i)})}\right)^2 + \frac{10}{n}f_{pen}(\vec{x}) + f_{opt}$	$-5 \leq x_i \leq 5$
	$w_i = \begin{cases} 10 & i = 1 \\ 1.1 + 8\frac{i-2}{99} & i > 1 \end{cases}$	
	$\vec{y}^{(i)} = \begin{cases} \text{vector of length } n \text{ uniformly drawn from } [-4,4]^n & i = 1 \\ \text{vector of length } n \text{ uniformly drawn from } [-4.9,4.9]^n & i > 1 \end{cases}$	
Rotated Rosenbrock*	$\text{rosenbrock}(\mathbf{R}(\vec{x}))$	$-2.048 \leq x_i \leq 2.048$
Rotated Rastrigin*	$\text{rastrigin}(\mathbf{R}(\vec{x}))$	$-5.12 \leq x_i \leq 5.12$
Rotated Schwefel*	$\text{schwefel}(\mathbf{R}(\vec{x}))$	$-512.03 \leq x_i \leq 511.97$

TABLE I
TEST PROBLEMS USED IN THE STUDY.

* indicates a minimization problem: the study negates all minimization problems to make them maximization problems for consistency. The input vector to each function if $\vec{x}$, and is of length n. $\mathbf{R}$ is an orthonormal rotation matrix, computed as described in Section III. $\mathbf{\Lambda}$ is a diagonal matrix with constant entries. $\mathbf{C}_i$ is a diagonal matrix with random entries $c \in [0,1]$. T_{osz} is a small oscillatory perturbation. f_{opt} is a small x offset for the optimum value and f_{pen} is an edge penalty function.

fitness individuals are more likely to be generated in the future (and in some cases, low fitness individuals are generated less often). A simplifying approach to the very difficult task of generating a high-dimensional joint distribution is to break the joint distribution into separate univariate distributions for each gene. Implementations include Univariate Marginal Distribution Algorithm (UMDA) [3], the Compact Genetic Algorithm (CGA) [4], and Population Based Incremental Learning (PBIL) [5]. These are often referred to as *univariate* or *marginal* EDAs.

By projecting the high dimensional distribution into multiple marginal distributions, univariate EDAs ignore linkages between genes, resulting in potentially poor performance on non-separable problems. To remedy this, various *multivariate* EDAs have been proposed that include information about the linkages between genes. These multivariate EDAs represent the linkages explicitly as joint distributions in various ways. The extended compact GA (ECGA) [6] establishes joint distributions among subsets of variables. The Iterated Density Estimation Evolutionary Algorithm (IDEA) [7] builds explicit conditional distributions across the variable space. The Bayesian Optimization Algorithm (BOA) [8], and Hierarchical BOA (hBOA) [9], represent linkages between variables as Bayesian networks, with various compact local representations of joint distributions.

Recent theoretical connection has been established between univariate EDAs and CCEAs [10] which opens up the possibility of using ideas from cooperative coevolution to improve univariate EDAs. The CCEA community has long understood that these (essentially identical) performance difficulties are due to certain game-theoretic anomalies arising from the multiple optimization procedures occurring in parallel along each margin [11]. Rather than establish multivariate relationships, CCEA theory instead suggests assessing individuals using the maximum over multiple *trials* with other collaborating individuals to form joint solutions [1]. In basic test problems, it has been shown that performance improves with more trials, and with sufficiently large numbers of trials the CCEA reduces to a kind of an EA (lacking the pathological conditions). Trials are expensive, however, leading to various approaches which maintain archives of high-quality collaborators [1], [12], [13] in order to reduce the number of trials necessary to assess fitness well. The EDA literature rarely attempts this approach, but the connection between the two suggests that it might be a viable alternative to moving to multivariate algorithms.

Most of these results have been with regard to a small number of subpopulations (usually two). But recent trends in CCEAs have increased the number of subpopulations to ten [14], [15], thirty [16], or even one thousand [17]. Our recent work [18] has recently shown that for complex problems, large numbers of subpopulations are indeed often preferable.

III. TEST FUNCTIONS

Table I shows the problems used in this paper, chosen for their different traits:

- *Rastrigin*, *Schwefel*, *Rosenbrock*, *Weierstrass* and *Gallagher's Gaussian 101-me Peaks* (hereafter *GG101me*) are standard functions. Weierstrass and Gallagher's may be both found in the Black Box Optimization Baseline 2009 (BBOB 09) [19]. Note that Rastrigin is additively separable.
- *Rotated Rastrigin*, *Rotated Schwefel* and *Rotated Rosenbrock* are randomly rotated versions of these functions, in order to increase the linkage between variables. A single fixed $m \times m$ rotation matrix $\mathbf{R}$ is used for all problems. The rotation matrix was generated in a fashion similar to that described in [20]: first, all entries in $\mathbf{R}$ were produced using random standard-normal Gaussian noise. Then for i from 1 to m, each row $\mathbf{R}_i$ was decreased by $\sum_{j=i}^{m}\langle \mathbf{R}_i\ \mathbf{R}_j\rangle \mathbf{R}_j$, then renormalized.

Target Function	*Subpops*	*Trial Performance (Best → Worst)*
Rosenbrock	2	$\overline{4.1\ 1.1\ 2.1}\ \overline{2.2\ 4.2\ 8.2}\ \overline{8.4\ 4.4\ 8.8}\ 8.1$
	8	$\overline{8.1\ 4.1\ 2.1}\ 1.1\ \overline{8.2\ 4.2\ 2.2}\ \overline{8.4\ 4.4}\ 8.8$
	32	$\overline{8.1\ 4.1}\ \overline{2.1\ 8.2}\ \overline{4.2\ 1.1}\ 2.2\ 8.4\ 4.4\ 8.8$
	128	$\overline{8.1\ 4.1}\ \overline{8.2}\ 2.1\ \overline{4.2\ 1.1}\ 2.2\ 8.4\ 4.4\ 8.8$
Rastrigin	2	$\overline{1.1\ 2.1\ 4.1\ 8.1}\ \overline{4.2\ 8.2\ 2.2}\ \overline{4.4\ 8.4}\ 8.8$
	8	$\overline{8.1\ 4.1\ 2.1}\ 1.1\ \overline{8.2\ 4.2\ 2.2}\ \overline{8.4\ 4.4}\ 8.8$
	32	$\overline{8.1\ 4.1}\ 2.1\ \overline{8.2\ 1.1}\ 4.2\ 2.2\ 8.4\ 4.4\ 8.8$
	128	$8.1\ 4.1\ 2.1\ 8.2\ \overline{1.1\ 4.2}\ 2.2\ 8.4\ 4.4\ 8.8$
Schwefel	2	$\overline{1.1\ \overline{2.1}\ 4.1\ 8.1}\ 2.2\ \overline{4.2\ 8.2}\ 4.4\ 8.4\ 8.8$
	8	$\overline{1.1\ \overline{2.1}\ 4.1\ 8.1}\ \overline{2.2\ 4.2}\ 8.2\ 4.4\ 8.4\ 8.8$
	32	$\overline{1.1\ 2.1\ 8.1\ 4.1}\ \overline{2.2\ 8.2\ 4.2}\ \overline{8.4\ 4.4}\ 8.8$
	128	$\overline{1.1\ 8.1\ 2.1\ 4.1}\ \overline{4.2\ 2.2\ 8.2}\ \overline{8.4\ 4.4}\ 8.8$
Rotated Rastrigin	2	$\overline{2.1\ 1.1}\ \overline{4.2\ 2.2}\ \overline{8.2\ 8.4\ 4.4}\ 8.8\ 4.1\ 8.1$
	8	$\overline{2.1\ 1.1\ \overline{4.2\ 2.2}\ \overline{8.4\ 4.4}\ 8.2}\ 8.8\ 4.1\ 8.1$
	32	$\overline{2.1\ 1.1\ 4.2\ 2.2\ 8.4\ 4.4\ \overline{8.2\ 8.8}}\ 4.1\ 8.1$
	128*	$4.1\ 2.1\ \overline{1.1\ 8.2\ \overline{8.1\ 4.2}\ 2.2\ 8.4\ 4.4\ 8.8}$
Rotated Schwefel	2	$4.2\ 2.2\ \overline{8.4\ \overline{4.4\ 1.1}\ 8.8\ \overline{8.2\ \overline{4.1\ 8.1\ 2.1}}}$
	8	$\overline{2.2\ 1.1}\ \overline{4.4\ \overline{8.1\ 2.1}\ \overline{4.2\ 4.1}\ \overline{8.8\ 8.4\ 8.2}}$
	32	$\overline{1.1\ 2.2}\ 4.4\ \overline{8.8\ \overline{2.1}\ 4.2\ 8.4\ 4.1\ 8.2\ 8.1}$
	128*	$\overline{4.4\ 2.1\ 4.1\ \overline{1.1\ 2.2\ 8.4\ 8.8\ 4.2\ 8.2}\ 8.1}$

TABLE II
SUMMARY OF CCEA CONVERGENCE EXPERIMENTS.

Trials are indicated with $k.d$ where k is the number of trials, and d is the divisor of the subpopulation size. Treatments are ordered from best to worst. Overbars indicate groups with statistically insignificant differences. * Indicates runs which were doubled in number of generations in order to determine convergence.

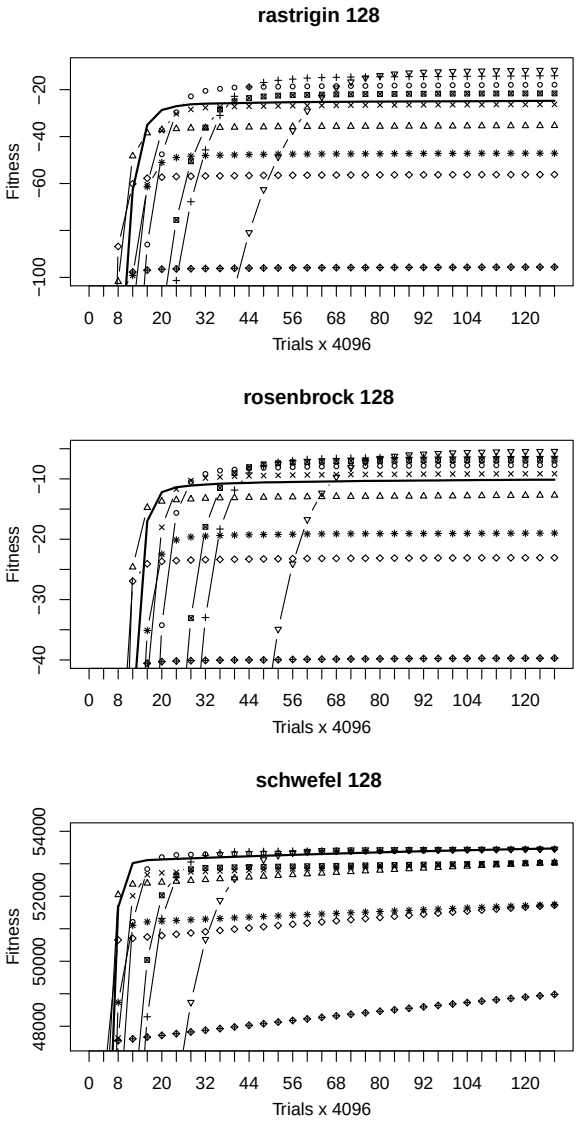


Fig. 1. Results for CCEA Experiment, Part I (128-subpopulation results shown). *Legend:* — 1.1 ◯ 2.1 △ 2.2 + 4.1 × 4.2 ◊ 4.4 ▽ 8.1 ☒ 8.2 ✳ 8.4 ⟠ 8.8

Rastrigin, Schwefel, Rosenbrock, Rotated Rastrigin, and Rotated Schwefel were used in the CCEA study. We then expanded on this study to add the Rotated Rosenbrock, Weierstrass, and Gallagher's functions to the Univariate EDA study.

IV. Cooperative Coevolutionary Algorithms

Our first goal was to determine if, and to what degree, multiple trials per fitness evaluation really benefited CCEAs for difficult and high-dimensional problems.

We used a CCEA with parallel update timing (that is, all subpopulations would be updated each generation). Breeding used tournament selection (of size 2), then one-point crossover, then Gaussian mutation ($\sigma = 0.01$), reselecting mutation until the new gene was within valid gene boundaries. The CCEA was tested using five problems, each with floating-point genotypes of size 128. Those problems were *Rosenbrock*, *Rastrigin*, *Rotated Rastrigin*, *Schwefel*, and *Rotated Schwefel*, as described in Table I. We gave each problem a fixed budget of $N = 1024^2$ trials, except in certain situations where we doubled the trials to examine convergence properties. We set a base population size $p = 1024$ and a base number of generations $g = 1024$.

Cooperative coevolution entails several parameters beyond standard evolutionary algorithms. In previous work [2] we examined the performance of these parameters, and our choices here are partly informed by them.

Update Timing: Should subpopulations be updated and bred in *parallel* or *sequentially*? CCEAs traditionally have been sequential, but we found that parallel update timing performs better (echoed by [21]).

Collaboration Scheme: How should collaborators be chosen? Traditionally CCEAs have used the best-performing previous individuals as collaborators, but we found that, except for very simple and linearly separable problems, random collaborators via *shuffling* performed better. We perform shuffling with parallel update timing as follows. Each generation the order of individuals in each subpopulation is randomly shuffled. Then for each i, individuals indexed i from each subpopulation

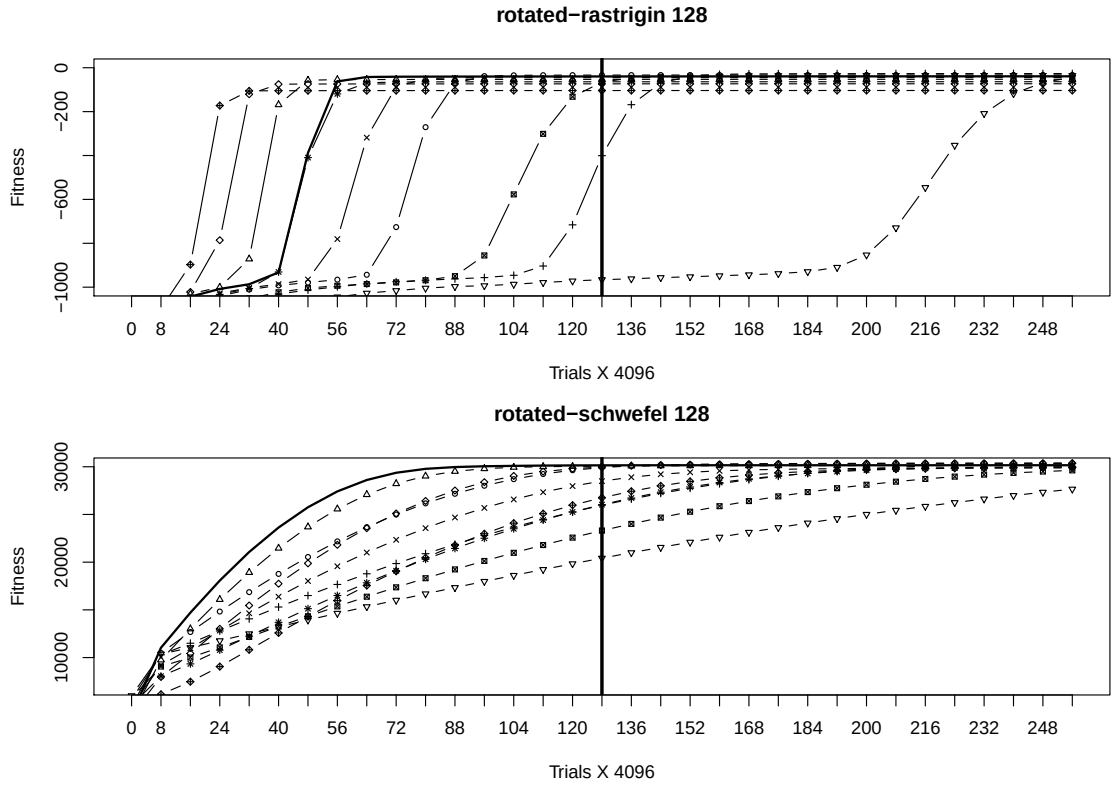


Fig. 2. Results for CCEA Experiment, Part II (128-subpopulation results only shown). The Rotated Rastrigin and Rotated Schwefel problems were run twice as long (only necessary for 128 subpopulations) to examine convergence. *Legend:* — 1.1 ◯ 2.1 △ 2.2 + 4.1 × 4.2 ◊ 4.4 ▽ 8.1 ☒ 8.2 ✳ 8.4 ⟠ 8.8

were tested together in a trial. Shuffling requires fewer trials per generation than other methods (the exact number is equal to the subpopulation size s). This permits different ways to distribute a budget of N trials. In our previous work, two such approaches performed particularly well, slightly modified here:

- Fix the subpopulation size s to p for each subpopulation, regardless of the number of subpopulations (q), and set the generations to g to keep the total trials to N. This had larger subpopulation sizes and shorter generations. We called this *Parallel Shuffled Pops* or *PP*, and found that it was favored by Rosenbrock, Rastrigin, Rotated Rastrigin, and Schwefel.
- Fix the total population size to $p \times 2$: each of q subpopulations would thus be of size $s = 2 \times p/q$. Set the generations to $g/2 \times q$, for a total of N trials. This had smaller subpopulation sizes and longer generations. We called this *Parallel Shuffled Gens* or *PG*, and found that it was favored by Rotated Schwefel.

Note that the two strategies (PP and PG) diverged fairly radically, in terms of distribution of trials, when the number of subpopulations was high: PP would produce large total population sizes, while PG would produce very long generations.

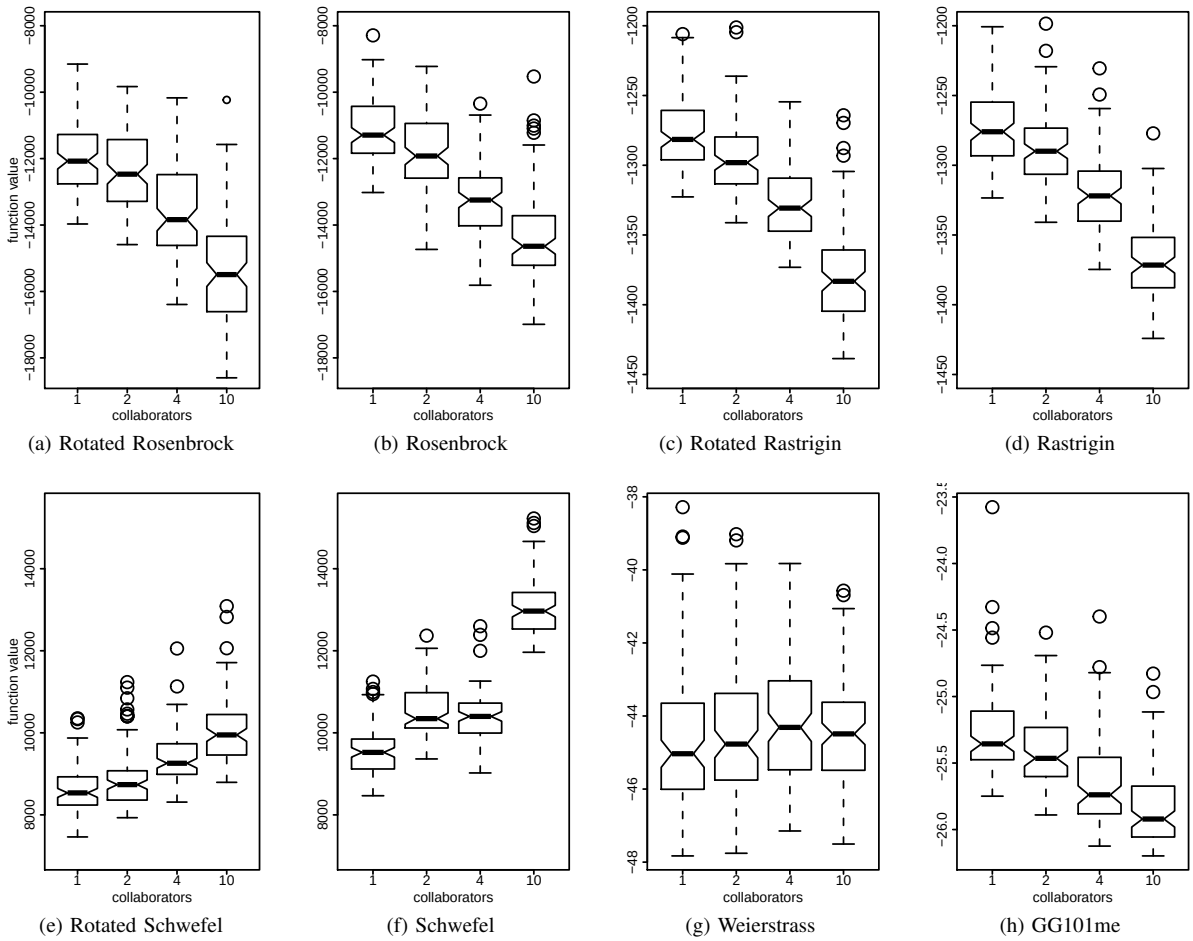
Number of Subpopulations: We found that more subpopulations generally results in more variance of performance: good-performing updating timing and collaboration scheme collaborations will perform even better, while poor-performing ones will perform even worse. This trend generally continues clear to 128 subpopulations (one subpopulation per gene), essentially in the realm of an EDA. Thus the most important result would be the 128 subpopulation result.

CCEA Experiments

We asked the following question: what happens when you set the fitness of an individual not based on a *single* trial but on the maximum over some k trials? We examined 1, 2, 4, and 8 trials per fitness assessment, performing multiple trials by simply shuffling multiple times.

To retain N total trials, we had to either reduce the subpopulation size, the number of generations, or both. This led to a second question: if you wish to increase evaluations, which should we reduce? We allocated these reductions using a subpopulation divisor (d). For two trials, we tried either halving the subpopulation size ("2.2", $d = 2$) or the generations ("2.1"). For four trials, we quartered the subpopulation size ("4.4", $d = 4$), quartered the generations ("4.1"), or divided each in half ("4.2"). For eight trials, we cut the subpopulation size to 1/8 ("8.8" $d = 8$), or the generations ("8.1"), or quartered the subpopulation size and halved the generations ("8.4"), or the opposite ("8.2").

We tested with 2, 8, 32, and 128 subpopulations, and performed 100 individual runs per treatment. Statistical signifi-



(a) Rotated Rosenbrock (b) Rosenbrock (c) Rotated Rastrigin (d) Rastrigin

(e) Rotated Schwefel (f) Schwefel (g) Weierstrass (h) GG101me

Fig. 3. EDA Experimental Results at 100,000 Evaluations

cance was tested using ANOVAs with a Bonferroni correction of $p = 1/400$ to guarantee a p-value of $1/20$ for the CCEA experiments as a whole.

Some techniques take longer to converge, but ultimately produce a better result. We were interested in both situations though the second is the only one easily quantifiable. We began by performing the runs until the runs had largely converged. For the Rotated Schwefel and Rotated Rastrigin, this required doubling the generations as shown in Figure 2.

We had predicted that multiple trials would be of benefit particularly for problems with high degree of linkage (such as the rotated problems). But our results were somewhat otherwise.

Convergence Results: Table II shows the convergence results of these runs: the crucial lines are those with 128 subpopulations. As can be seen, for Rosenbrock and Rastrigin, it was clearly advantageous to perform more trials and to distribute those trials such that subpopulation sizes were kept as high as possible (that is, 2.1, 4.1, and 8.1). Schwefel was similar, but in all cases a single trial (1.1) was as good as any other.

The rotated problems, where we had expected more trials to be helpful, didn't show this as much as expected. Rotated Rastrigin preferred multiple trials with large subpopulation sizes (4.1, 2.1), but a single trial (1.1) was not far behind. Interestingly, the results for < 128 subpopulations placed 4.1 and 8.1 dead last; 128 resulted in a big jump for them.

Rotated Schwefel preferred a different technique (PG versus PP). As expected, methods preserving generation lengths (4.2, 2.2, 4.4, etc.) often tended to perform better than they had in the past. However, 1.1 was crucially often just as good as any other.

Time to Convergence: Convergence is one thing: but what if your budget is less than a million trials? We wondered what techniques took longer to converge.

Figures 1 and 2 show the results for 128 subpopulations: other figures are similar. These figures made it very clear: methods with more trials per evaluation took longer, often much longer, to converge, if they maintained larger subpopulation sizes. Specifically, the slowest converging methods were (starting with the slowest) 8.1, 4.1, 8.2, and usually 2.1 and 4.2. In contrast, 1.1 was almost always one of the very fastest converging methods. The only methods which were generally as fast as 1.1 (or in a few cases faster) were 8.8, 4.4, and 2.2. These methods maintained long generation lengths.

Many such techniques simply have a slower C-like curve: but Rotated Rastrigin consistently produced S-like curves which were essentially shifted in time, making the convergence differences very clear.

In short: methods with more evaluations and which maintained large subpopulation sizes often performed best. But they took a long time, sometimes a *very* long time, to converge to those best-performing results. Surprisingly, in the rotated problems multiple evaluations were less helpful.

V. Univariate Estimation of Distribution Algorithms

Following the lines of our CCEA experiments, we wanted to determine what happens when the univariate EDA distributions are updated based on multiple samples rather than a single sample.

Our EDA of choice was PBIL [5], modified to operate on real-valued functions. We used a mixture of three Guassians (described below) to initialize each marginal distribution, per [22]. In addition, we added shuffling in a manner similar to the CCEA experiments, as described later.

The basic PBIL operates as follows:

```
1: for each of g generations do
2:     for each of p candidate solutions do
3:         for each variable do
4:             Sample from the distribution
5:         Assemble candidate solution
6:         Evaluate the candidate solution
7:     Select highest valued candidate solutions
8:     Update variable distributions
```

In each generation, PBIL generates a total of p individuals as follows: First, it generates p samples from the distribution for each variable. For each i from 1 to p, it then groups together the sample indexed i from each variable and forms an individual. All p individuals are then assigned a fitness value. The least fit individuals are discarded via truncation selection (we discarded 2/3 of them). We updated the distribution for each variable using the variable value from each of the remaining individuals.

Real-valued Extensions via Mixture of Gaussians: The basic form of PBIL was originally designed for discrete functions. The extension to real-valued functions alters the nature of the variable used, and the distribution used to model the variable. In this implementation, we used initialized the distribution for each variable using a mixture of three Gaussians providing coverage across the full range of the variable [1]. At each generation, each variable's distribution was updated following [22].

Shuffling: PBIL is amenable to shuffling in a fashion reminiscent of our approach to CCEAs. Ordinarily each of the p samples of each variable would be used in a single trial. But we can provide multiple trials per sample, reducing the number of samples accordingly. To implement shuffling, we modified the basic PBIL algorithm so that it reduces the number of samples generated per variable by a factor d.

This was done as follows. Each generation PBIL must generate a total of p individuals. To do so, we generated s samples from each variable. Then $d = p/s$ times we performed the following two steps. First, for each variable, we randomly shuffled the order of its s samples. Second for each i from 1 to s, we grouped together the sample indexed i from each variable to form an individual. All told we generated $p = d \times s$ individuals.

PBIL would then test all p individuals. Each sample would be tested in d trials as a member of some individual. PBIL would then proceed as usual.

Because each sample could appear in d candidate solutions, a sample represented in multiple highly fit candidate solutions might be included in the update sample set as many as d times. If a sample was represented in multiple candidate solutions of poor fitness, it might not be represented in the update sample set at all.

[1]For a range (a, b), our initial distribution was $N(a+(b-a)/\varphi, (b-a)/4)$, $N(b-(b-a)/\varphi, (b-a)/4)$, and $N(a+(a+b)/2, (b-a)/3)$, where φ is the golden mean ≈ 1.61803.

Target Function	*Number of Trials*
Rotated Rosenbrock	1 2 4 10
Rosenbrock	1 2 4 10
Rotated Rastrigin	1 2 4 10
Rastrigin	1 2 4 10
Rotated Schwefel	10 4 2 1
Schwefel	10 $\overline{2\ 4}$ 1
Weierstrass	$\overline{4\ 2\ 10\ 1}$
Gallagher's Gaussian 101-me Peaks	1 2 4 10

TABLE III
SUMMARY OF EDA EXPERIMENTS.

The treatments are ordered in decreasing performance. $\overline{\text{Overbars}}$ indicate statistically insignificant differences.

EDA Experiments

We tested our estimation of distribution algorithms using all the non-trial problems in Table I with floating-point representation. Each problem had $g = 1000$ generations and a PBIL population size $p = 100$. To generate 100 individuals per generation, the modified PBIL would perform shuffling 1, 2, 4, or 10 times: this would result in generating either 100, 50, 25, or 10 samples per marginal distribution. Thus the tradeoff here was number of samples (essentially "individuals") per generation versus the number of trials per sample.

We performed 100 runs per treatment, and computed statistics using a Wilcoxon rank-sum test at a 95% confidence level. We expanded on the CCEA set of target functions, adding Rotated Rastrigin, Weierstrass, and Gallagher's Gaussian 101-me Peaks.

Similar to CCEAs, we expected that increasing the number of trials would improve performance in some situations (or at least not hurt it). However this was only the case for a few problems. As summarized in Figure 3, and elaborated in Figure 4, Schwefel and Rotated Schwefel were the only two problems to improve performance with increased number of trials. Statistical significance results are shown in Table III. Other than Weierstrass, increasing the number of trials changed performance: but only occasionally in a positive direction.

We note that Schwefel and Rotated Schwefel were the primary methods in the study in which optima were far from one another, in the corners of the space rather than near the center. Were more evaluations helping reduce miscoordination among the variables in these problems?

Note that the absolute performance is significantly different than our CCEA results. This is largely due to the order of magnitude fewer trials in this study.

VI. Conclusions

CCEAs have often used multiple trials to compensate for the reduced information resulting from projection of the joint space into individual marginal subspaces. Most work in this area has been in small numbers of subpopulations and relatively simple problems. We performed a study which expanded this to large numbers of subpopulations and complex problems, using a shuffling technique which we have found in the past to be best performing. We found that increasing the numbers of trials per fitness evaluation, at the expense of shorter generations, will improve converged results to some degree: but will also slow convergence, and so may not be worthwhile.

Because they too rely on marginal subspaces, Univariate EDAs can be shuffled in a manner similar to CCEAs. We modified a real-valued version of PBIL accordingly to see if it would have similar results. But it it did not. In two problems (Schwefel and Rotated Schwefel) multiple evaluations helped: but in the others multiple evaluations did nothing or, more often than not, hurt matters.

Because the number of samples changes with shuffling, in many ways the PBIL approach taken here is roughly equivalent to the CCEA 2.2, 4.4, and 8.8 methods; likewise except for Schwefel it performed very similarly to them. As future work we wish to further modify the PBIL approach to reduce generations but retain sample counts, to more closely approximate the CCEA 2.1, 4.1, and 8.1 approaches which seem to have performed better.

VII. Acknowledgements

We thank Marc Shoenauer at the Institut National Recherche en Informatique et en Automatique (INRIA) for supporting insight into the implementation of rotations in BBOB.

This research was supported by NFS Grant 0916870.

References

[1] L. Panait, "The analysis and design of concurrent learning algorithms for cooperative multiagent systems," Ph.D. dissertation, George Mason University, Fairfax, Virginia, 2006.

[2] S. Luke, K. Sullivan, and F. Abidi, "Large scale empirical analysis of cooperative coevolution," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2011.

[3] H. Muehlenbein and G. Paass, "From recombination of genes to the estimation of distributions: I. binary parameters," in *Parallel Problem Solving form Nature (PPSN VI)*, 1996.

[4] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE-EC*, vol. 3, no. 4, p. 287, Nov. 1999.

[5] S. Baluja, "Population-Based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon University, School of Computer Science, Tech. Rep. CS-94-163, Jun. 1994.

[6] G. Harik, "Linkage learning via probabilistic modeling in the ECGA," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Tech. Rep. 99010, 1999.

[7] P. A. N. Bosman and D. Thierens, "Expanding from discrete to continuous estimation of distribution algorithms: The IDEA," in *Parallel Problem Solving form Nature (PPSN VI)*, 2000, pp. 767–776.

[8] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Linkage problem, distribution estimation, and bayesian networks," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Tech. Rep. 98013, 1998.

[9] M. Pelikan and D. E. Goldberg, "Hierarchical bayesian optimization algorithm = bayesian optimization algorithm + niching + local structures," in *Optimization by Building and Using Probabilistic Models (OBUPM)*, 2001, pp. 217–221.

[10] C. Vo, L. Panait, and S. Luke, "Cooperative coevolution and univariate estimation of distribution algorithms," in *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms (FOGA)*, 2009, p. 141.

[11] R. P. Wiegand, "An analysis of cooperative coevolutionary algorithms," Ph.D. dissertation, George Mason University, 2004.

[12] A. Bucci and J. B. Pollack, "On identifying global optima in cooperative coevolution," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2005.

[13] E. D. de Jong, "Towards a bounded pareto-coevolution archive," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2004.

[14] T. Ray and X. Yao, "A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2009, pp. 983–989.

[15] Z. Yang, J. Zhang, K. Tang, X. Yao, and A. C. Sanderson, "An adaptive coevolutionary differential evolution algorithm for large-scale optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2009, pp. 102–109.

[16] C. H. Yong and R. Miikkulainen, "Cooperative coevolution of multi-agent systems," Department of Computer Sciences, The University of Texas at Austin, Tech. Rep. AI07-338, 2001.

[17] W. Chen, T. Weise, , Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Proceedings of Parallel Problem Solving from Nature*, 2011.

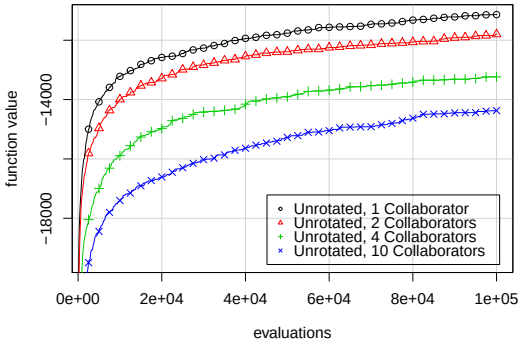
[18] S. Luke, K. Sullivan, and F. Abidi, "Large scale empirical analysis of cooperative coevolution," Department of Computer Science, George Mason University, Tech. Rep. GMU-CS-TR-2011-2, 2011.

[19] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2009.

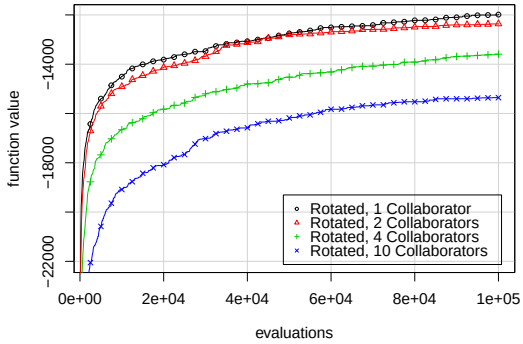
[20] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[21] E. Popovici and K. D. Jong, "Sequential versus parallel cooperative coevolutionary algorithms for optimization," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2006, pp. 1610–1617.

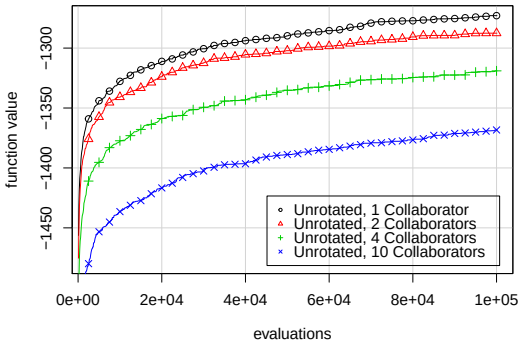
[22] M. Gallagher, M. Frean, and T. Downs, "Real-valued evolutionary optimization using a flexible probability density estimator," in *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO)*. Morgan Kaufmann, 1999, pp. 840–846.



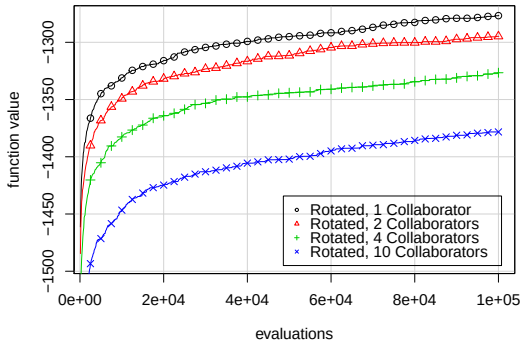
(a) Rosenbrock



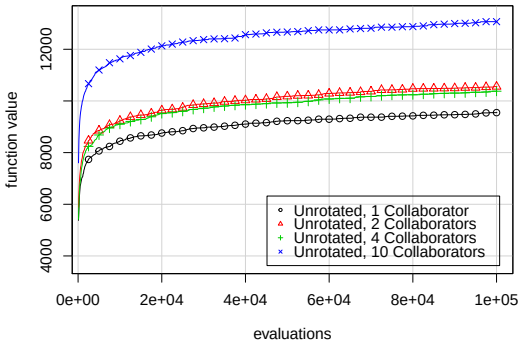
(b) Rotated Rosenbrock



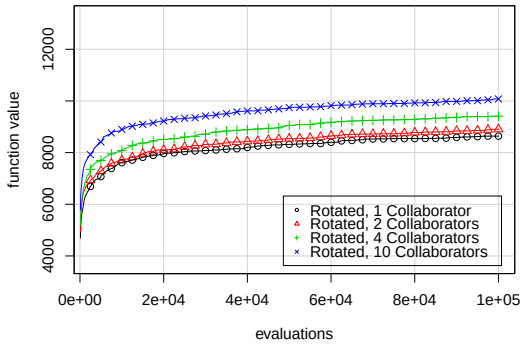
(c) Rastrigin



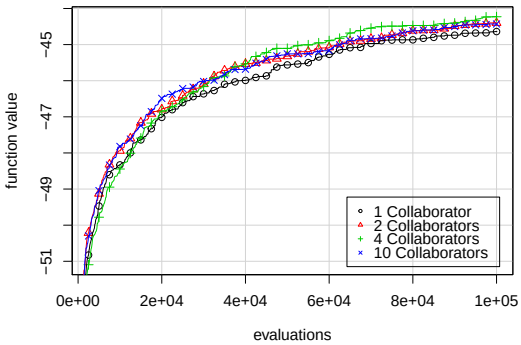
(d) Rotated Rastrigin



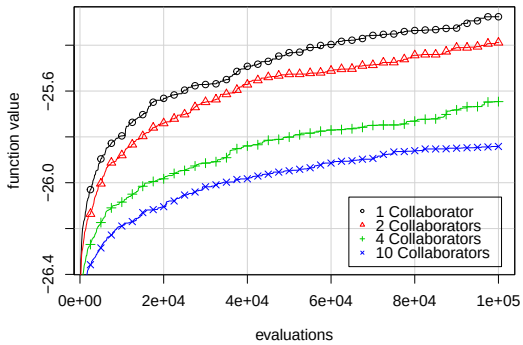
(e) Schwefel



(f) Rotated Schwefel



(g) Weierstrass



(h) GG101me

Fig. 4. EDA Experimental Results