Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2014, Article ID 182973, 10 pages
http://dx.doi.org/10.1155/2014/182973

Research Article

A Comparative Study of EAG and PBIL on Large-Scale Global Optimization Problems

Imtiaz Hussain Khan

Department of Computer Science, King Abdulaziz University, Jeddah, P.O. Box 80200, Saudi Arabia

Correspondence should be addressed to Imtiaz Hussain Khan; ihkhan@kau.edu.sa

Received 16 June 2014; Revised 13 November 2014; Accepted 17 November 2014; Published 7 December 2014

Academic Editor: Sebastian Ventura

Estimation of Distribution Algorithms (EDAs) use global statistical information effectively to sample offspring disregarding the location information of the locally optimal solutions found so far. Evolutionary Algorithm with Guided Mutation (EAG) combines global statistical information and location information to sample offspring, aiming that this hybridization improves the search and optimization process. This paper discusses a comparative study of Population-Based Incremental Learning (PBIL), a representative of EDAs, and EAG on large-scale global optimization problems. We implemented PBIL and EAG to build an experimental setup upon which simulations were run. The performance of these algorithms was analyzed in terms of solution quality and computational cost. We found that EAG performed better than PBIL in attaining a good quality solution, but the latter performed better in terms of computational cost. We also compared the performance of EAG and PBIL with MA-SW-Chains, the winner of CEC'2010, and found that the overall performance of EAG is comparable to MA-SW-Chains.

1. Introduction

Many search and optimization techniques have been developed to solve complex optimization problems, like travelling salesman problem. One widely studied approach in this area is Estimation of Distribution Algorithms (EDAs) [1–3]. The main difference between traditional evolutionary algorithms [4–6], for example, genetic algorithms, and EDAs lies in their offspring generation strategies. Traditional evolutionary algorithms use crossover and mutation to generate new solutions, whereas EDAs use probabilistic models to sample offspring. The probabilistic models are based on global statistical information, extracted from population. According to proximate optimality principle [7], which assumes that good solutions have similar structure, an ideal offspring generator should be able to generate a solution that is close to the best solutions found so far. In this respect, both evolutionary algorithms and EDAs have their own merits and demerits. Evolutionary algorithms allow that the new solutions would not be far away from the best solutions found so far, whereas EDAs have no mechanism to directly control the similarity between an offspring and its parent. On the other hand, EDAs better control the similarity among solutions in the current population because they use the global statistical information effectively to sample offspring.

Evolutionary Algorithm with Guided Mutation (EAG) [8] combines global statistical information (i.e., the EDA approach) and location information (i.e., the traditional evolutionary algorithmic approach) to sample offspring. The authors evaluated the performance of EAG on maximum clique problem and showed promising results. In [9], Khan used numerical optimization functions [10, 11] and standard genetic algorithm test problems [12, 13] to compare the performance of EAG against two classic EDAs, Population-Based Incremental Learning (PBIL) [14] and Compact Genetic Algorithm [15]. The previous studies revealed that EAG performed better than its competitors. However, so far, the performance of EAG was measured on small-scale optimization problems. Therefore, it is unclear to us how scalable EAG is.

In this study, we evaluated thoroughly the performance of EAG against PBIL on a set of benchmark functions for large-scale global optimization problems [16]. The results of these two algorithms are also compared with MA-SW-Chains [17], the winner of CEC'2010 competition [16]. This paper contributes in two different ways. First, our results

Require:
α (learning rate), μ (mutation rate), δ (mutation shift),
l (length of the probability vector), and N (population size)
(1) Initialize the probability vector p
(2) Repeat Steps 3 to 7 until stopping criteria are met
(3) Generate a population of N solutions using p
(4) Evaluate the fitness of the solutions (generated in Step 3)
(5) Select the k best solutions from N
(6) Update the probability vector p as following:

$$p^{t+1} = (1 - \alpha)\, p^t + \alpha \left(\frac{1}{k} \sum_{i=1}^{k} b_i^t \right)$$

(7) Mutate the probability vector p as following:
if random$(0, 1) < \mu$ then

$$p^{t+1} = (1 - \delta)\, p^t + \delta \cdot b^t$$

Pseudocode 1: The PBIL algorithm.

further strengthen the previous findings that a combination of EDAs and traditional evolutionary algorithms works better than standalone EDAs or evolutionary algorithms. Second, our findings indicate that, originally developed for discrete optimization problems, EAG is also suitable for continuous optimization problems. The algorithm is scalable and its performance is comparable to MA-SW-Chains, the winner of CEC'2010.

The rest of this paper is organized as follows. In Section 2, we give a background overview of PBIL, EAG, other closely related EDA approaches, and large-scale global optimization problems. The empirical study is outlined in Section 3. We discuss main findings and limitations of our approach in Section 4. The paper concludes in Section 5.

2. Background

2.1. PBIL. PBIL is a statistical approach to evolutionary computation in which solutions are represented as fixed length binary strings $b = (b_1, b_2, b_3, \ldots, b_n)$. A probability vector $p = (p_1, p_2, p_3, \ldots, p_n)$ is used to sample offspring. In this vector, p_i measures the distribution of 1 s (and consequently 0 s) at the ith position of the simulated population in a given search space. Initially, these probabilities are set to 0.5 at each position to give a uniform distribution over the search space. Solutions are generated using p; for each solution b a 1 is generated at position b_i with probability p_i. The probabilities in p are then moved gradually towards 1 or 0 as the search progresses. The PBIL algorithm is illustrated in Pseudocode 1.

Apart from stopping criteria and the number of best solutions (k) used to update the probability vector, PBIL is sensitive to four main parameters: learning rate (α), mutation rate (μ), mutation shift (δ), and population size (N). Among these, α and δ are kept *low* (e.g., 0.1 and 0.02, resp.) [14].

2.2. EAG. EAG (Evolutionary Algorithm with Guided Mutation) is a hybrid of evolutionary algorithms and EDAs. Variation operators in evolutionary algorithms directly use the location information of the locally optimal solutions found so far, disregarding the distribution of promising solutions in the search space. The offspring thus produced are close to their parents, but they may be far away from other best solutions in the current population. This is because evolutionary algorithms do not benefit from the global statistical information. On the other hand, EDAs use the global statistical information effectively to sample offspring, but they disregard the location information of the optimal solutions found so far. This is an important limitation in EDAs because there is no mechanism to directly control the similarity between the new solutions and the current "good" solutions. EAG combines global statistical information and the location information of optimal solutions to sample offspring, aiming at the fact that this hybridization would improve the solution quality. A new variation operator "guided mutation" is developed in EAG. The pseudocode of EAG is similar to PBIL except that the solutions for the next generation are produced using the guided-mutation operator. The guided-mutation operator is illustrated in Pseudocode 2.

The guided-mutation operator is sensitive to the guided-mutation rate β. The operator decides on the basis of β to sample new offspring by copying the location information either from the parent or from the probability vector p; with the larger value of β, more genes of the offspring (y) are sampled from the probability vector. EAG is sensitive to learning rate (λ), guided-mutation rate (β), and population size (N) [8].

2.3. Other Related Approaches. A large body of work has been published on EDAs [1–3, 14, 15, 18–26]. Here we review a piece of work, which is devoted to hybrid approaches in EDAs [18, 21–25]; for a detailed review on EDAs, please see [27]. Mahnig and Mühlenbein used a hybrid approach in which mutation operators were introduced into EDAs using Bayesian prior [18]. They found that the introduction of mutation in EDAs greatly decreases the dependence of an optimal population size. In another study, Handa incorporated mutation operators into EDAs to maintain the diversities in the EDA population [21]. His results show that mutation improves the search ability of EDAs, even with a small population size. In [23], Santana et al. combined

```
Require:
β (guided-mutation rate),
p = p_1, p_2, ..., p_l (probability vector),
    b = b_1, b_2, ..., b_l (solution vector, b_i ∈ 0, 1)
for i = 1 to l do
   h = random(0, 1)
   if β > h          // sample ith bit from the probability vector, p
      if p[i] > h
         y[i] = 1
      else
        y[i] = 0
   else              // copy ith bit from the parent, b
      y[i] = b[i]
end_for
```

Pseudocode 2: The guided-mutation operator.

EDAs with the variable neighborhood search (VNS) heuristic and found that this hybrid approach performed reasonably well as compared to simple EDA or simple VNS approaches. It is worth mentioning here that the problem dimensions explored in these studies were relatively small; for example, in [21] the problem dimensions were not more than 70. In yet another study [24], Valdez et al. developed a hybrid algorithm that combines EDA with support vector machine for selection of key feature genes. They compared their method with some other hybrid EDAs and found it effective. In a very recent study [25], the authors presented an EDA based on Gaussian probability distribution. They showed that on higher dimension problems their algorithm offered better performance than its competitors.

2.4. Large-Scale Global Optimization. In any empirical investigation of evolutionary algorithms, the selection of test problems is always vital. Care must be taken to try and select problems that will hopefully prove illuminating for the investigation at hand. As a rule of thumb, at least two factors are often considered while selecting test problems: (a) comparison with the previous findings/results and (b) representativeness. Usually, a well-studied and broad range of problems are suitable because they can provide useful means of comparison with the previous experimental results.

In recent years, evolutionary computation community has seen a substantial number of studies to evaluate the performance of evolutionary algorithms on large-scale global optimization problems, such as those with more than one hundred decision variables [16, 28]. These test problems pose significant challenges not only because of their high dimensionality, that is, "curse of dimensionality" [29], but also because most of them are *nonseparable*. A function of n variables is separable if it can be rewritten as a sum of n functions of justone variable.

In this study, we compared the performance of PBIL and EAG on the benchmark functions provided for CEC'2010 [16]. Their characteristics vary from separable to nonseparable and unimodal to multimodal functions. Moreover, all these functions are shifted, scalable, and minimization problems, whose global minimum value is known, which is 0. These functions are provided in Table 1; for details, see [16].

3. Empirical Study

We used an empirical approach to compare the strength of EAG and PBIL on large-scale global optimization problems. Three primary aims were pursued in this study: (a) experimental setup upon which simulations are run, (b) optimal parameter settings for EAG and PBIL (formative experiment), and (c) statistical analysis to compare the performance of EAG with PBIL on the selected test problems (summative experiment). The performance of each algorithm was analyzed in terms of solution quality and total elapsed time. In what follows, we discuss the experimental setup, the optimal parameter settings, and the analysis of results in turn.

3.1. Experimental Setup. An empirical investigation requires a system upon which experiments are run. We implemented the system in Matlab-R2009a. A brief description of this implementation is outlined as follows.

(i) The solutions were encoded as binary strings. The length of a string was set to 10 times the dimension of the test problem (in this study, for the summative experiment, the dimension of a problem was set to 1000 for each test function).

(ii) To keep uniformity, the probability vector was initialized to 0.5 for both PBIL and EAG.

(iii) Initial population was sampled randomly for both algorithms using the same initial probability vector; the population size was kept 100 throughout the study.

(iv) Because fitness of an individual is computed from its phenotype value, a function *binry2real* was implemented which maps binary value (genotype) into the corresponding decimal value (phenotype).

(v) In each generation, *best* and *average* fitness of the population were recorded; for each algorithm the total *time* that elapsed was also recorded.

Table 1: Properties of the benchmark functions CEC'2010 [16].

Function	Modality	Separability	Domain
F_1: shifted elliptic function	Unimodal	Separable	$[-100, 100]^D$
F_2: shifted Rastrigin's function	Multimodal	Separable	$[-5, 5]^D$
F_3: shifted Ackley's function	Multimodal	Separable	$[-32, 32]^D$
F_4: single-group shifted and m-rotated elliptic function	Unimodal	Single-group m-nonseparable	$[-100, 100]^D$
F_5: single-group shifted and m-rotated Rastrigin's function	Multimodal	Single-group m-nonseparable	$[-5, 5]^D$
F_6: single-group shifted and m-rotated Ackley's function	Multimodal	Single-group m-nonseparable	$[-32, 32]^D$
F_7: single-group shifted and m-rotated Schwefel's problem 1.2	Unimodal	Single-group m-nonseparable	$[-100, 100]^D$
F_8: single-group shifted and m-rotated Rosenbrock's function	Multimodal	Single-group m-nonseparable	$[-100, 100]^D$
F_9: $(D/2m)$-group shifted and m-rotated elliptic function	Unimodal	$(D/2m)$-group m-nonseparable	$[-100, 100]^D$
F_{10}: $(D/2m)$-group shifted and m-rotated Rastrigin's function	Multimodal	$(D/2m)$-group m-nonseparable	$[-5, 5]^D$
F_{11}: $(D/2m)$-group shifted and m-rotated Ackley's function	Multimodal	$(D/2m)$-group m-nonseparable	$[-32, 32]^D$
F_{12}: $(D/2m)$-group shifted and m-rotated Schwefel's problem 1.2	Unimodal	$(D/2m)$-group m-nonseparable	$[-100, 100]^D$
F_{13}: $(D/2m)$-group shifted and m-rotated Rosenbrock's function	Multimodal	$(D/2m)$-group m-nonseparable	$[-100, 100]^D$
F_{14}: (D/m)-group shifted and m-rotated elliptic function	Unimodal	(D/m)-group m-nonseparable	$[-100, 100]^D$
F_{15}: (D/m)-group shifted and m-rotated Rastrigin's function	Multimodal	(D/m)-group m-nonseparable	$[-5, 5]^D$
F_{16}: (D/m)-group shifted and m-rotated Ackley's function	Multimodal	(D/m)-group m-nonseparable	$[-32, 32]^D$
F_{17}: (D/m)-group shifted and m-rotated Schwefel's problem 1.2	Unimodal	(D/m)-group m-nonseparable	$[-100, 100]^D$
F_{18}: (D/m)-group shifted and m-rotated Rosenbrock's function	Multimodal	(D/m)-group m-nonseparable	$[-100, 100]^D$
F_{19}: shifted Schwefel's problem 1.2	Unimodal	Fully nonseparable	$[-100, 100]^D$
F_{20}: shifted Rosenbrock's function	Unimodal	Fully nonseparable	$[-100, 100]^D$

Both algorithms terminated when the maximum number of generations exceeded a preset limit, which was set to 10000 generations for the formative experiment and $3.0e+06$ generations for the summative experiment.

3.2. Formative Experiment. Before running the experiment properly, we conducted a pilot study to find the optimal parameter values for EAG and PBIL. Since parameter tuning in an evolutionary algorithms-based system is a challenging task, a principled approach is required for this purpose. In this study, we were interested in tuning three parameters for EAG, *learning rate* (λ), *guided-mutation rate* (β), and *population size* (N), and four parameters for PBIL, *learning rate* (α), *mutation rate* (μ), *mutation shift* (δ), and *population size* (N). We used the following numerical optimization functions to find the optimal parameter values for each algorithm; for details, see [30]. (We encoded these functions as 300-bit strings, 10 consecutive bits for each dimension.)

(i) *Sphere Function*. This is a smooth, unimodal function. It is separable and relatively easy to optimize.

(ii) *Rosenbrock's Function*. This is a complex optimization function, which follows a parabolic trajectory.

(iii) *Rastrigin's Function*. This function has many local minima, but only one global minimum.

(iv) *Griewank's Function*. This is a multimodal function with an exponentially increasing number of local minima as its dimension increases.

(v) *Ackley's Function*. This is a multimodal function. It is nonseparable and difficult to optimize.

```
EAG
for N from 10 to 100 Step 10
  for λ from 0.1 to 1.0 Step 0.1
    for β from 0.1 to 1.0 Step 0.1
      Record solution-fitness value
    end_for
  end_for
end_for
PBIL
for N from 10 to 100 Step 10
  for α from 0.1 to 1.0 Step 0.1
    for μ from 0.01 to 0.1 Step 0.01
      for δ from 0.01 to 0.1 Step 0.01
        Record solution-fitness value
      end_for
    end_for
  end_for
end_for
```

Algorithm 1: Parameter tuning mechanism for EAG and PBIL.

To find the optimal values for the selected sensitive parameters, we changed them in an orderly manner and recorded the fitness value as shown in Algorithm 1.

We ran both algorithms on each test problem in turn for a maximum of 10000 generations; 10 independent runs were performed to stabilize the parameter settings. With regard to EAG, we observed that a smaller value of λ (mostly 0.1), a larger value of β (mostly 0.9), and a larger value of N (mostly $N > 50$) provided better results. We further fine-tuned the parameter β; keeping $N = 100$ and $\lambda = 0.1$,

it was observed that $\beta = 0.95$ gave much better result; on this combination of parameter values, EAG was able to find the global optimal value for each test function. Similarly, for PBIL, it was observed that a combination of parameter values ($\mu = 0.02$, $\delta = 0.05$, $\alpha = 0.1$, and $N = 100$), the same as reported in [14], gave better results. Therefore, in the summative experiment, we used $N = 100$, $\lambda = 0.1$, and $\beta = 0.95$ as parameter values for EAG (throughout this study, for EAG, we used *mutation rate* = 0.02, *mutation shift* = 0.05, and *negative learning rate* = 0.075 [8]). Also, for PBIL, the following combination of parameter values was used: $\alpha = 0.1$, $\mu = 0.02$, $\delta = 0.05$, and $N = 100$.

3.3. Summative Experiment. The performance of an evolutionary algorithm can be manifested in different ways. We are interested in two performance gains: (a) *quality improvement* and (b) *speed improvement*. However, we mainly focused on the former. This is important, because even though most evolutionary algorithms use sophisticated strategies to find good solutions, finding an acceptably "good-enough" solution is not guaranteed. So, if the solution quality is not "good-enough" then the secondary aspects such as speed are of little consequence. We define that an algorithm A beats an algorithm B under quality criterion if algorithm A attains a converged solution of higher fitness than algorithm B. Similarly, an algorithm A beats an algorithm B under speed criterion if algorithm A attains a solution of a given quality/fitness in lesser time than algorithm B. It is possible, however, that a gain in quality may be obtained at the expense of time and vice versa. Therefore, in order to prove that the overall performance of an algorithm A is *better* than B, we must show one of the following propositions to be true.

P1: algorithm A performs better than algorithm B in both speed and quality.

P2: algorithm A performs better in terms of speed without being outperformed in quality.

P3: algorithm A performs better in terms of quality without being outperformed in speed.

To test these propositions, a series of experiments were run on a Matlab-R2009a system, Sony Vaio Core i5-2430M, with 2.4 GHz speed and 4 GB RAM (DDR3). We recorded (a) the fitness of a solution for each generation, (b) the best fitness by the end of a run, (c) the best fitness at various points ($5e+05$, $1e+06$, $1.5e+06$, $2e+06$, and $2.5e+06$) during the evolution process, and (d) the total time that elapsed.

3.4. Results and Discussion. In this section, we present the simulation results of PBIL and EAG on each test problem. To test whether the apparent differences in the performance gain are statistically significant, we also report on a two-tailed pairwise t-test.

We measured the quality of a solution x in terms of function error value defined as $f(x) - f(x^*)$, where x^* is the known global optimum of f [31]. Table 2 shows the simulation results of the two algorithms on each test problem in 25 independent runs, including the overall best solution (Best), the mean of best-of-run solution averaged over 25 runs (Mean), the standard deviation in best-of-run solution (σ), and the P value for a two-tailed t-test (the P value measures whether or not the pairwise difference in the best solution for each algorithm in 25 independent runs is statistically significant; moreover, in this study, $P < 0.05$ is our standard value for statistical significance).

It is clear from Table 2 that both algorithms failed to find the known global optimum value for any function. However, the performance of EAG looks better than PBIL on all functions. It is interesting to note that on separable functions, namely, F_1, F_2, and F_3, both algorithms offered competitive performance. In the remaining 17 functions, F_4–F_{20}, the performance of EAG was far better than PBIL.

Figure 1 depicts the evolutions of solutions for some select functions. It is clear from the figures that EAG performed better than PBIL during the search process. It was observed that, for functions F_4–F_{20}, the average solution quality of EAG at various points ($5e+05$, $1e+06$, $1.5e+06$, $2e+06$, and $2.5e+06$) was greatly improved over PBIL.

We also compared the performance of EAG and PBIL with MA-SW-Chains [17], the winner of CEC'2010 [16]. It is evident from Table 2 that MA-SW-Chains performed better than EAG and PBIL. However, interestingly, we found that, for two separable functions (F_1 and F_2) and seven partially nonseparable functions (F_4–F_9 and F_{13}), the best results of EAG are better than the best mean results of MA-SW-Chains (cf. Table 2, the results in bold).

A one-way ANOVA was used to test whether the apparent differences in the best solutions among the competing algorithms are significant or not. Results are shown in Table 3. Results differed significantly across the three algorithms on 20 select functions: $F(2, 57) = 7.46$, F-critical = 2.83, $P < 0.5$. To further analyze the performance of EAG and PBIL, we report pairwise comparisons using a t-test. Pairwise comparisons revealed that EAG outperformed PBIL on 17 functions (F_4–F_{20}): $P < 0.01$. On the remaining three functions (F_1–F_3), the difference between EAG and PBIL was not significant though: $P > 0.5$.

To test our second performance criterion (speed gain), we recorded time (in milliseconds) taken by each algorithm to finish the search process. As discussed earlier, both PBIL and EAG terminated after a fixed number of generations, which were set to $3.0e + 6$. Again, all the simulation results were averaged over 25 independent runs. The results are shown in Table 4, which indicate that PBIL converged more quickly than EAG on all functions. Again, a one-way ANOVA test revealed that the speed differences between EAG and PBIL are statistically significant: $F(1, 38) = 5.93$, F-critical = 2.09, $P < 0.5$ (Table 5). Pairwise comparisons further revealed that the convergence time of PBIL was significantly faster ($P < 0.01$) than EAG on all functions. These results suggest that EAG is computationally more expensive than PBIL.

None of our propositions (P1, P2, and P3) was found true, because EAG performed better than PBIL in terms of solution quality, but the latter outperformed the former in terms of speed. But as mentioned earlier, in this study, solution quality was our primary objective; therefore we can conclude that EAG offered better performance than PBIL. To

Table 2: The solution quality of PBIL and EAG on CEC'2010 benchmark functions.

	PBIL		EAG		MA-SW-Chains	
	Mean (σ)	Best	Mean (σ)	Best*	Mean (σ)	Best
F_1	4.93e − 02 (1.97e − 01)	3.58e − 02	4.74e − 02 (1.82e − 01)	3.47e − 02	2.10e − 14 (1.99e − 14)	3.18e − 15
F_2	9.69e + 02 (3.08e + 02)	9.17e + 02	9.03e + 02 (1.16e + 02)	7.95e + 02	8.1e + 02 (5.88e + 01)	7.04e + 02
F_3	2.14e + 00 (2.35e + 00)	1.97e + 00	1.83e + 00 (1.16e + 00)	1.25e + 00	7.28e − 13 (3.44e − 13)	3.34e − 13
F_4	9.38e + 11 (5.47e + 11)	8.24e + 11	3.91e + 11 (1.52e + 10)	**3.49e + 11**	3.53e + 11 (3.12e + 10)	3.04e + 11
F_5	5.85e + 09 (3.84e + 08)	5.03e + 09	4.29e + 08 (1.37e + 07)	**4.02e + 07**	1.68e + 08 (1.04e + 08)	2.89e + 07
F_6	7.96e + 05 (5.03e + 04)	4.73e + 05	4.87e + 04 (1.83e + 03)	**1.05e + 04**	8.14e + 04 (2.84e + 04)	8.13e − 07
F_7	3.73e + 04 (3.92e + 04)	1.19e + 04	5.62e + 02 (4.94e + 01)	**4.17e + 01**	1.03e + 02 (8.70e + 01)	3.35e − 03
F_8	5.42e + 08 (9.83e + 07)	1.48e + 08	4.38e + 07 (5.96e + 07)	**4.31e + 06**	1.41e + 07 (3.68e + 07)	1.54e + 06
F_9	9.31e + 07 (3.15e + 07)	4.32e + 07	6.25e + 07 (2.81e + 07)	**1.03e + 07**	1.41e + 07 (1.15e + 06)	1.19e + 07
F_{10}	8.69e + 03 (3.72e + 02)	5.87e + 03	8.63e + 03 (1.02e + 02)	2.47e + 03	2.07e + 03 (1.44e + 02)	1.81e + 03
F_{11}	6.93e + 02 (2.95e + 01)	6.28e + 02	9.75e + 01 (9.92e + 00)	7.52e + 01	3.80e + 01 (7.35e + 00)	2.74e + 01
F_{12}	7.61e + 01 (4.18e + 01)	7.29e + 01	8.06e + 00 (3.27e + 00)	7.92e + 00	3.62e − 06 (5.92e − 07)	2.65e − 06
F_{13}	5.17e + 04 (1.09e + 03)	1.26e + 04	4.25e + 03 (1.16e + 03)	**4.07e + 02**	1.25e + 03 (5.72+02)	3.86e + 02
F_{14}	5.92e + 10 (1.24e + 09)	5.83e + 09	8.73e + 07 (1.02e + 07)	7.59e + 07	3.11e + 07 (1.93e + 06)	2.79e + 07
F_{15}	7.74e + 03 (2.91e + 03)	7.68e + 03	7.31e + 03 (1.17e + 02)	3.46e + 03	2.74e + 03 (1.22e + 02)	2.56e + 03
F_{16}	3.49e + 03 (1.52e + 02)	6.22e + 02	2.47e + 02 (8.09e + 01)	1.02e + 02	9.98e + 01 (1.40e + 01)	8.51e + 01
F_{17}	2.18e + 03 (1.14e + 02)	2.07e + 03	3.84e + 01 (1.72e + 01)	1.69e + 01	1.24e + 00 (1.25e − 01)	1.04e + 00
F_{18}	1.93e + 06 (1.03e + 05)	1.87e + 06	7.62e + 04 (2.08e + 03)	7.58e + 03	1.30e + 03 (4.36e + 02)	7.83e + 02
F_{19}	5.27e + 07 (3.81e + 06)	4.93e + 07	3.15e + 06 (1.77e + 05)	8.37e + 05	2.85e + 05 (1.78e + 04)	2.49e + 05
F_{20}	8.74e + 04 (1.39e + 04)	6.04e + 04	4.07e + 04 (1.84e + 03)	3.99e + 03	1.07e + 03 (7.29e + 01)	9.25e + 02

*The boldface values indicate that the *best* results of EAG are better than the *best mean* results of MA-SW-Chains.

Table 3: One-way ANOVA test results (solution quality).

Source of variation	SS	df	MS	F	P value	F-crit.
Between groups	8.68E + 19	2	4.34E + 19	7.46	0.34	2.83
Within groups	8.48E + 21	57	1.49E + 20			
Total	8.56E + 23	59				

this endeavor, we also explored another dimension of speed gain, the first ever generation in which the best solution was found. Interestingly, it was observed that overall EAG found the best solution earlier than PBIL; these results are also apparent in Figure 1.

4. General Discussion

The study presented here revealed two primary results. First, EAG outperformed PBIL in attaining a good-quality solution. This indicates that, in sampling offspring, a combination of

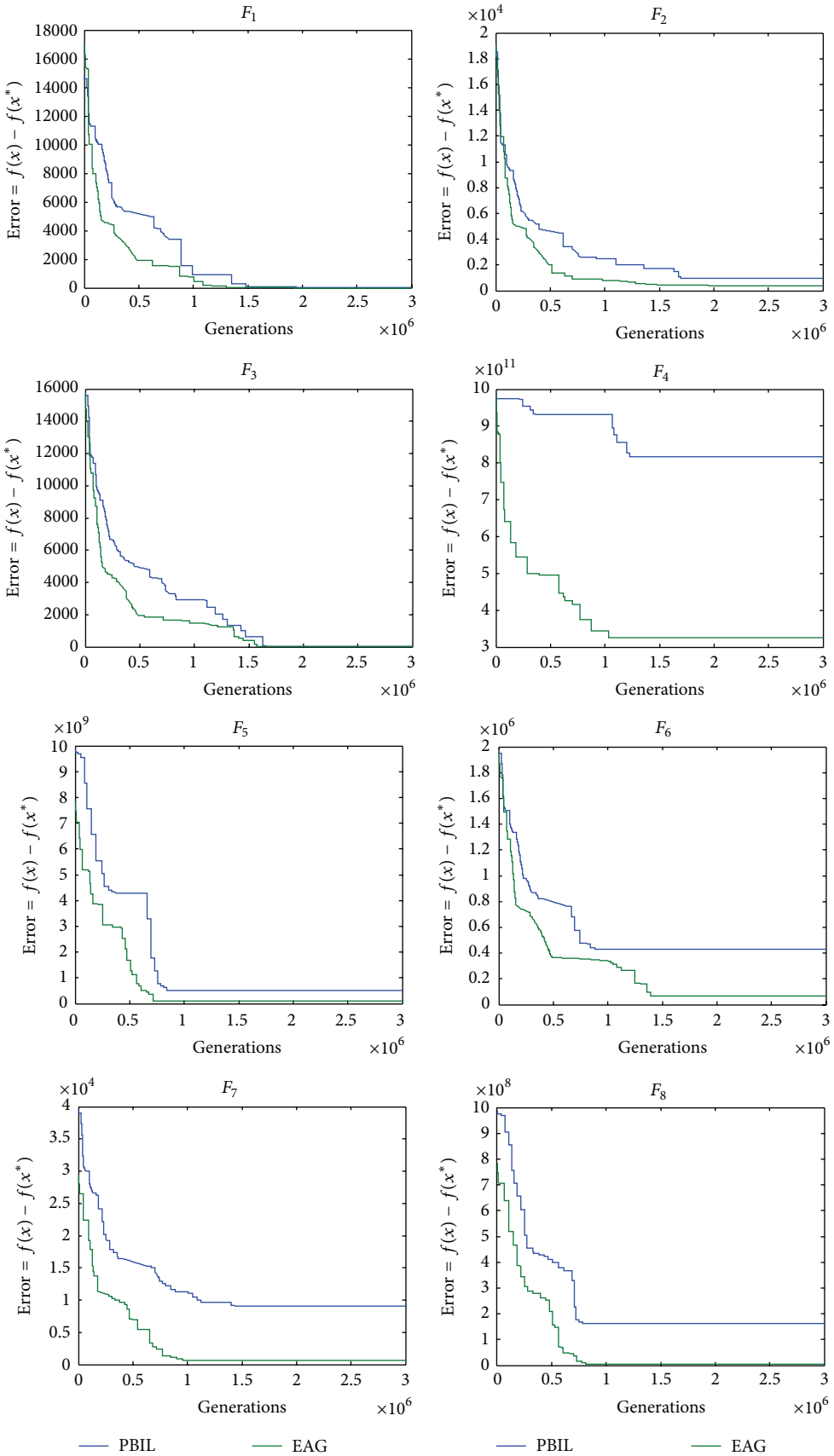

Figure 1: Continued.

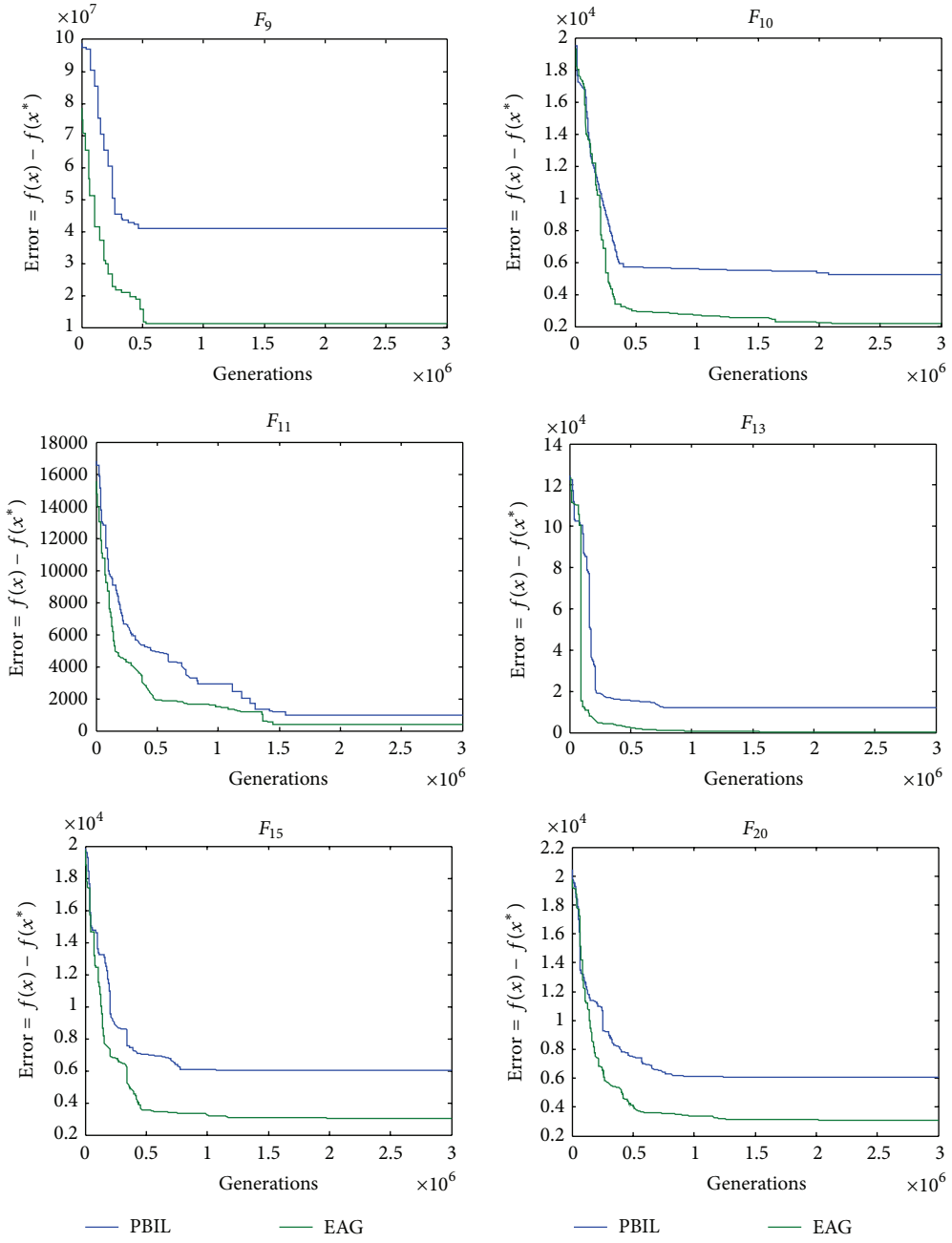

FIGURE 1: Convergence graph of PBIL and EAG on the select functions: F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}, F_{11}, F_{13}, F_{15}, and F_{20}.

global statistical information and the location information of the solutions found so far is better than using global statistical information only. We observed that, on separable functions, both algorithms offered competitive performance but on nonseparable functions EAG outperformed PBIL. This suggests that the underlying assumption in EDAs that problem variables are independent may prevent efficient convergence to the global optimum when problem variables interact strongly.

Second, PBIL was found faster than EAG. This suggests that EAG is computationally expensive. The reason behind this expensiveness is that the *guided-mutation operator* used in EAG involves many computations in sampling offspring. We also observed that the speed of EAG becomes further slow with the increase in chromosome length.

If both solution quality and computational time are addressed, this raises the question of how these two dimensions should be traded off against each other. If the output of one algorithm was better than that of another but was found more slowly, which of the two algorithms should be preferred? Perhaps solution quality should be given more weight as compared to speed.

Finally, the rather high difference between the known global optimum solution and the solution found by PBIL and

Table 4: Computational time of PBIL and EAG on CEC'2010 benchmark functions.

	PBIL		EAG		P value
	Time (ms)	σ	Time (ms)	σ	
F_1	3.21e + 07	2.42e + 01	4.96e + 07	1.37e + 01	<0.01
F_2	3.51e + 08	3.17e + 01	4.13e + 08	9.16e + 01	<0.01
F_3	5.27e + 07	6.73e + 01	5.71e + 07	7.73e + 01	<0.01
F_4	2.05e + 10	3.12e + 02	2.82e + 10	2.39e + 02	<0.01
F_5	3.87e + 08	1.01e + 02	4.12e + 08	1.17e + 02	<0.01
F_6	6.13e + 08	5.72e + 01	8.07e + 08	7.26e + 01	<0.01
F_7	8.58e + 08	6.36e + 01	9.93e + 08	9.17e + 01	<0.01
F_8	1.26e + 09	7.19e + 01	3.86e + 09	8.44e + 01	<0.01
F_9	3.91e + 07	2.49e + 01	6.52e + 7	4.79e + 01	<0.01
F_{10}	1.63e + 07	3.27e + 01	3.74e + 07	6.31e + 01	<0.01
F_{11}	7.18e + 06	2.19e + 01	9.07e + 06	2.75e + 01	<0.01
F_{12}	4.99e + 08	5.06e + 01	8.18e + 08	6.53e + 01	<0.01
F_{13}	5.77e + 08	4.96e + 01	7.84e + 08	7.19e + 01	<0.01
F_{14}	7.28e + 08	3.97e + 01	9.27e + 08	5.02e + 01	<0.01
F_{15}	2.37e + 07	1.73e + 01	3.26e + 07	2.08e + 01	<0.01
F_{16}	8.46e + 06	3.26e + 01	9.38e + 06	3.91e + 01	<0.01
F_{17}	3.52e + 08	4.05e + 01	5.27e + 08	4.69e + 01	<0.01
F_{18}	4.21e + 09	6.71e + 01	4.73e + 09	9.37e + 01	<0.01
F_{19}	3.94e + 08	5.28e + 01	7.51e + 08	7.18e + 01	<0.01
F_{20}	5.69e + 07	3.16e + 01	7.13e + 07	4.29e + 01	<0.01

Note. It is important to mention here that we do not include time for MA-SW-Chains, because the authors did not report on computational time.

Table 5: One-way ANOVA test results (computational cost).

Source of variation	SS	df	MS	F	P value	F-crit.
Between groups	3.96E + 18	1	3.96E + 18	5.93	<0.01	2.09
Within groups	1.14E + 21	38	2.99E + 19			
Total	1.14E + 21	39				

EAG could be because of our *suboptimal* encoding scheme. We encoded the solutions (genotype) as bitstrings, but fitness of these solutions was computed from their phenotype values. Therefore, binary (genotype) to decimal (phenotype) conversion was necessary, and this conversion could have resulted in sufficient accuracy loss. It was also observed that this conversion takes a significant amount of time and as a whole degrades the performance of an algorithm in terms of speed as well.

5. Conclusion

This paper has described a comparative study of EAG and PBIL on large-scale global optimization problems. In a nutshell, we found that combining global statistical information and the location information of the solutions found so far significantly improves the quality of search/optimization process. We observed that, on separable functions, both algorithms offered competitive performance but on nonseparable functions EAG outperformed PBIL. These findings suggest that the location information should be used and it is not sufficient to use a very limited number of dependence relationships (i.e., the EDAs approach) to solve optimization and search problems. We also observed that EAG achieved better solution quality at the expense of more computational cost. We conclude that the computational overheads should not penalize EAG, because if the solution quality is not good-enough (e.g., in a fault-critical situation) then the secondary aspects such as speed are of little consequence.

The results show that, originally developed for discrete optimization problems, EAG is also suitable for continuous optimization problems. We found that the solution quality of EAG is comparable to MA-SW-Chains, the winner of CEC'2010.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by a King Abdulaziz University (KAU) funding (Grant no. 611-009-D1433). The author thanks KAU for their financial support. The author also thanks the editor and anonymous reviewers for their valuable comments.

References

[1] H. Mühlenbein and G. Paaß, *From Recombination of Genes to the Estimation of Distributions I. Binary Parameters*, pp. 178–187, Springer, 1996.

[2] S. Baluja and S. Davies, "Fast probabilistic modeling for combinatorial optimization," in *Proceedings of the 15th National Conference on Artificial Intelligence*, pp. 469–476, 1998.

[3] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: a New Tool for Evolutionary Computation*, Kluwer Academic Publishers, Boston, Mass, USA, 2001.

[4] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, Germany, 1973.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[6] J. R. Koza and R. Poli, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Complex Adaptive Systems, MIT Press, Cambridge, Mass, USA, 1992.

[7] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Norwell, Mass, USA, 1998.

[8] Q. Zhang, J. Sun, and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 192–200, 2005.

[9] I. H. Khan, "A comparative study of evolutionary algorithms," *International Journal of Artificial Intelligence*, vol. 12, no. 1, pp. 1–17, 2014.

[10] X.-S. Yang, "Test problems in optimization," in *Engineering Optimization: An Introduction with Metaheuristic Applications*, X.-S. Yang, Ed., John Wiley & Sons, New York, NY, USA, 2010.

[11] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC-2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, Singapore, 2005.

[12] G. T. Reinelt, *Tsplib*, Universität Heidelberg, 1995, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.

[13] A. Scholl and R. Klein, "Bin-packing data-set 2 for BPP-1," 2003, http://www.wiwi.uni-jena.de/entscheidung/binpp/bin2dat.htm.

[14] S. Baluja, "Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning," Tech. Rep. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pa, USA, 1994.

[15] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," in *Proceedings of the IEEE Conference on Evolutionary Computation*, pp. 523–528, 1998.

[16] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Bench-mark functions for the CEC-2010 special session and competition on large scale global optimization," Tech. Rep., Nature Inspired Computation and Applications Laboratory, USTC, Hefei, China, 2010.

[17] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization," in *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI '10)*, 3160, p. 3153, Barcelona, Spain, July 2010.

[18] T. Mahnig and H. Mühlenbein, "Optimal mutation rate using Bayesian priors for estimation of distribution algorithms," in *Stochastic Algorithms: Foundations and Applications: International Symposium, SAGA 2001 Berlin, Germany, December 13–14, 2001 Proceedings*, K. Steinhöfel, Ed., vol. 2264 of *Lecture Notes in Computer Science*, pp. 33–48, Springer, Berlin, Germany, 2001.

[19] J. M. Peña, V. Robles, P. Larrañaga, V. Herves, F. Rosales, and M. S. Pérez, "GA-EDA: hybrid evolutionary algorithm using genetic and estimation of distribution algorithms," in *Proceedings of the 17th International Conference on Innovations in Applied Artificial Intelligence*, pp. 361–371, Springer, 2004.

[20] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 127–136, 2004.

[21] H. Handa, "Estimation of distribution algorithms with mutation," in *Evolutionary Computation in Combinatorial Optimization*, pp. 112–121, Springer, Berlin, Germany, 2005.

[22] A. Ochoa and M. Soto, "Linking entropy to estimation of distribution algorithms," in *Towards a New Evolutionary Computation*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds., vol. 192 of *Studies in Fuzziness and Soft Computing*, pp. 1–38, Springer, Berlin, Germany, 2006.

[23] R. Santana, P. Larrañaga, and J. A. Lozano, "Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem," *Journal of Heuristics*, vol. 14, no. 5, pp. 519–547, 2008.

[24] S. I. Valdez, A. Hernández, and S. Botello, "A Boltzmann based estimation of distribution algorithm," *Information Sciences*, vol. 236, pp. 126–137, 2013.

[25] L. Li, H. Chen, C. Liu et al., "A robust hybrid approach based on estimation of distribution algorithm and support vector machine for hunting candidate disease genes," *The Scientific World Journal*, vol. 2013, Article ID 393570, 7 pages, 2013.

[26] Q. Xu, C. Zhang, and L. Zhang, "A fast elitism Gaussian estimation of distribution algorithm and application for PID optimization," *The Scientific World Journal*, vol. 2014, Article ID 597278, 14 pages, 2014.

[27] S. H. Chen, P. C. Chang, and Q. Zhang, "A self-guided genetic algorithm for flowshop scheduling problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 471–478, Trondheim, Norway, May 2009.

[28] K. Tang, X. Yao, P. N. Suganthan et al., "Benchmark functions for the CEC-2008 special session and competition on large scale global optimization," Tech. Rep., Nature Inspired Computation and Applications Laboratory, USTC, Beijing, China, 2007.

[29] R. E. Bellman, *Dynamic Programming*, Dover Books on Mathematics, Dover Publications, Princeton, NY, USA, 2003.

[30] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC-2005 special session on real-parameter optimization," Tech. Rep., Nanyang Technological University, Singapore, 2005.

[31] J. Brest, A. Zamuda, I. Fister, and M. S. Maučec, "Large scale global optimization using self-adaptive differential evolution algorithm," in *Proceedings of the 6th IEEE World Congress on Computational Intelligence (WCCI '10)*, Barcelona, Spain, July 2010.