A Comparative Study of Evolutionary Algorithms

Imtiaz Hussain Khan

Department of Computer Science
Faculty of Computing and Information Technology
King Abdulaziz University, Saudi Arabia
ihkhan@kau.edu.sa

ABSTRACT

This article describes a comparative study of Evolutionary Algorithm with Guided mutation (EA/G) against Population-Based Incremental Learning (PBIL) and Compact Genetic Algorithm (CGA). Both PBIL and CGA are representatives of Estimation of Distribution Algorithms (EDAs), a class of algorithms which uses the global statistical information effectively to sample offspring disregarding the location information of the local optimal solutions found so far. On the other hand, EA/G uses global statistical information as well as location information to sample offspring. We implemented the algorithms to build an experimental setup upon which simulations were run. The performance of the algorithms was analyzed on numerical function optimization problems and standard genetic algorithm test problems in terms of solution quality and computational cost. We found that EA/G outperformed PBIL and CGA in attaining a good-quality solution, but both PBIL and CGA performed better than EA/G in terms of computational cost.

Keywords: Evolutionary Algorithms, Estimation of Distribution Algorithms, Guided Mutation, Optimization Functions

1998 ACM Computing Classification System: I.2.8, I.2.m, I.6.6.

1 Introduction

Finding the global optimum solution for hard optimization problems (i.e., the problems which involve combinatorial explosion in the solution space, e.g., a traveling salesman problem) has always been a challenge. A variety of search and optimization techniques have been developed for solving such problems, and a promising approach has been the development of evolutionary algorithms (EAs) (Rechenberg 1973, Holland 1992, Koza & Poli 1992). EAs are stochastic and iterative search methods, which take their inspiration from natural selection and evolution. Unlike traditional optimization techniques (e.g., gradient descent) that operate on a single solution at a time, EAs are population-based heuristics, meaning that they operate on a set of solutions in the search space. In this search space, each solution is assigned a fitness value which quantifies how the solution is. The search process, which is guided by the fitness of the solutions, proceeds as follows. EAs start by creating a number of (candidate) solutions to form an initial population. Then, a competitive selection is made to choose parents, from which new solutions (offspring) are produced using reproduction/variation operators. Finally, again, a competitive selection is made from the old population and the offspring to form the new population of the next generation. This process is repeated until some predefined termination criteria are satisfied. At the heart of EAs, there are two variation operators: crossover and mutation, which act directly on selected solutions and allow diversity in the search space. To sample offspring, EAs directly use the location information of the local optimal solutions (i.e., the actual positions of these solutions) in the search space. However, they do not take the interactions among the variables/genes (which characterize the distribution of promising solutions in the search space) into account. Estimation of Distribution Algorithms (EDAs) (Mühlenbein & Paaß 1996, Baluja & Davies 1998, Larrañaga & Lozano 2002) are also stochastic search methods, but unlike traditional EAs, they represent search space by maintaining explicit probabilistic models of promising candidate solutions. These probabilistic models are based on the global statistical information, which is extracted from the current population. EDAs use this information effectively to sample offspring. However, they do not use the information about the locations of the local optimal solutions found so far to guide the search. Evolutionary Algorithm with Guided Mutation (EA/G) (Zhang, Sun & Tsang 2005) combines global statistical information (i.e., the EDA approach) and the location information (i.e., the traditional EA approach) to sample offspring.

In this article, we empirically compared[1] EA/G against Population-Based Incremental Learning (PBIL) (Baluja 1994) and Compact Genetic Algorithm (CGA) (Harik, Lobo & Goldberg 1999); the latter two algorithms are representatives of EDAs. The performance of these algorithms was evaluated on numerical function optimization functions (Yang 2010, Suganthan, Hansen, Liang, Deb, Chen, Auger & Tiwari 2005) and standard genetic algorithm test problems (Reinelt 1995, Scholl & Klein 2003).

The rest of this article is organized as follows. Section 2 gives a background overview of EDA approaches and EA/G. The test problems are discussed in Section 3. The empirical study is outlined in Section 4. Results are discussed in Section 5. The article concludes in Section 6.

2 Background

2.1 EDA Approaches

A large body of work has been published on EDAs (Baluja 1994, Mühlenbein & Paaß 1996, Harik, Goldberg & Miller 1997, Baluja & Davies 1998, Larrañaga & Lozano 2002, Peña, Robles, Larrañaga, Herves, Rosales & Pérez 2004, Zhang & Mühlenbein 2004, Chen, Chang & Zhang 2009). Here we review two EDAs: Population-Based Incremental Learning algorithm (PBIL) (Baluja 1994) and Compact Genetic Algorithm (CGA) (Harik et al. 1999). PBIL is a statistical approach to evolutionary computation in which solutions are represented as fixed length binary strings $b = (b_1, b_2, b_3, ..., b_n)$. A probability vector $p = (p_1, p_2, p_3, ..., p_n)$, where p_i measures the distribution of 1s (and consequently 0s) in the i^{th} position of the simulated population in a given search space, is used to sample offspring. 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 as follows. For each solution b, a 1 is generated at position b_i with probability p_i.The probabilities

[1]To the best of our knowledge, these algorithms have not been compared before.

in p are then moved gradually towards 1 or 0 as the search progresses. Because PBIL uses the best-fit individuals of the population to update the probability vector, it is expected that after a number of generations the probability vector will be shifted towards good solutions. It is important to mention here that, unlike evolutionary algorithms, in PBIL, mutation is applied on the probability vector rather than the solution itself.
Compact Genetic Algorithm (CGA), inspired by random-walk model (Harik et al. 1997), is also an EDA approach. The algorithm manages its population as a probability distribution over a set of solutions. Like PBIL, CGA also uses a probability vector, whose length is equal to the length of the chromosome. The value at a particular index in this probability vector measures the proportion of alleles (0 or 1) in the respective locus of the simulated population. In each generation, two candidate solutions are generated and then evaluated to determine the best of the two solutions (winner). If at any particular position the winner's bit is different from the loser's bit, the corresponding probability vector entry is shifted by 1/*N* (where *N* represents the theoretical population size) towards the winner bit.
Some researchers have proposed hybrid approaches to evolutionary algorithms (Zhang et al. 2005, Handa 2005, Ochoa & Soto 2006, Santana, Larrañaga & Lozano 2008, Niño, Ardila, Donoso & Jabba 2010, Niño 2012). Handa (2005) incorporated mutation operators into EDAs to maintain diversities in population. The author has shown that mutation improves the search ability of EDAs, even with a small population size. In another study, Santana et al. (2008) combined EDAs with the Variable Neighborhood Search (VNS) heuristic and found that this hybrid approach performed reasonably well as compared to simple EDA or VNS approaches. In yet another study, Niño (2012) proposed a hybrid approach based on automata theory, simulated annealing and evolutionary algorithms. The author has shown that the proposed approach not only avoids the local minima/maxima problem, it can also avoid finding infeasible solutions. The results of these studies suggest that hybrid approaches to EDAs improve the search performance. In what follows, we discuss Evolutionary Algorithm with Guided Mutation (EA/G) (Zhang et al. 2005), which is a hybrid approach.

2.2 Evolutionary Algorithm with Guided Mutation

Evolutionary Algorithm with Guided Mutation (EA/G) (Zhang et al. 2005) is a hybrid of EAs and EDAs. Variation operators in EAs directly use the location information of the local optimal solutions found so far, disregarding the distribution of promising solutions in the search space. The offspring thus produced may be very close to their parents, but there is no guarantee that they are also close to the other best solutions in the current population. This is because EAs 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 local 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. EA/G combines the global statistical information as well as the location information to sample offspring, aiming that this hybridization would improve the solution quality. A new variation operator "guided mutation" is proposed in EA/G. The pseudo-code of EA/G is similar to PBIL except that the solutions for the next generation are produced using the guided mutation operator. The pseudo-code for the guided mutation operator is shown below.

```
**Require**: $\beta$ (guided-mutation rate),
$p = p_1, p_2, ..., p_l$ (probability vector),
$b = b_1, b_2, ..., b_l$ (solution vector, $b_i \in 0, 1)$

*for* $i$ = 1 to $l$ *do*

    $h = random(0|1)$
    *if* $\beta > h$ // Sample the $i^{th}$ bit from $p$
        *if* $p[i] > h$, $y[i] = 1$
        *else* $y[i] = 0$
    *else* // Copy the $i^{th}$ bit from the parent
        $y[i] = b[i]$

*end for*
```

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

3 Test Problems

In any empirical investigation of EAs, 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, well studied and a broad range of problems are suitable because they can provide a useful means of comparison with the previous experimental results. Also, they can address the assumptions behind any research and therefore highlight particular strengths and weaknesses of the algorithms under scrutiny.
In this study, we are examining the performance of EA/G against PBIL and CGA on numerical optimization functions (Yang 2010, Suganthan et al. 2005) and standard genetic algorithm test problems (Reinelt 1995, Scholl & Klein 2003). It is instructive to give a brief overview of these test problems, before discussing the experiment.

3.1 Numerical Optimization Problems

There are many test functions in literature to test the performance of evolutionary algorithms. We have chosen the following numerical optimization functions (see Yang (2010) and Suganthan et al. (2005) for a detailed description of the functions $F_1 - F_5$ and $F_6 - F_{12}$, respectively), which have widely been used in the previous studies on evolutionary algorithms. (In this study, the number of dimensions (D) were set to 30 for each function; for shifted functions $z = x - o$, where o is the shifted global optimum.)

Sphere Function (F_1) is a smooth, uni-modal function. The function definition is given below.

$$f(x) = \sum_{i=1}^{D} {x_i}^2, x \in [-5.12, 5.12]^D \tag{3.1}$$

Rosenbrock's Function (F_2) is a complicated surface, which follows a parabolic trajectory and has a global optimum at just one point on the ridge. The function definition is given below.

$$f(x) = \sum_{i=1}^{D-1} ((x_i - 1)^2 + 100(x_{i+1} - x_i^2)^2), x \in [-5, 5]^D \tag{3.2}$$

Ackley's Function (F_3) is a complex multi-modal function. The function definition is given below.

$$f(x) = -20exp\Bigg(- 0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\Bigg) - exp\Bigg(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\Bigg) + 20 + e, x \in [-32.77, 32.77]^D \tag{3.3}$$

Rastrigin's Function (F_4) has many local minima, however, it has just one global minimum at the point [0, 0] in the x-y plane. At any local minimum other than [0, 0], the value of the function is greater than 0. The function definition is given below.

$$f(x) = 10D + \sum_{i=1}^{D} ({x_i}^2 - 10\cos(2\pi x_i)), x \in [-5.12, 5.12]^D \tag{3.4}$$

Griewank's Function (F_5) is a multi-modal function with an exponentially increasing number of local minima as its dimensions increase, making it very difficult to optimize. The function definition is given below.

$$f(x) = \frac{1}{4000}\sum_{i=1}^{D} {x_i}^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1, x \in [-600, 600]^D \tag{3.5}$$

Shifted Sphere Function (F_6) is a shifted, uni-modal, scalable and separable[2] function. The function definition is given below (where $bias = -450$).

$$f(x) = \sum_{i=1}^{D} {z_i}^2 + bias, x \in [-100, 100]^D \tag{3.6}$$

Shifted Rastrigin's Function (F_7) is a shifted, multi-modal, scalable and separable function. This function has many local minima, which makes it very difficult to find the global optimum value. The function definition is given below (where $bias = -330$).

$$f(x) = \sum_{i=1}^{D} ({z_i}^2 - 10\cos(2\pi z_i) + 10) + bias, x \in [-5, 5]^D \tag{3.7}$$

[2] A function of n variables is separable if it can be rewritten as a sum of n functions of just one variable, otherwise it is non-separable.

Shifted Schwefels's Function (F_8) is a shifted, uni-modal, scalable and non-separable function. The function definition is given below (where $bias = -440$).

$$f(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} {z_i}^2 + bias, x \in [-100, 100]^D \tag{3.8}$$

Shifted Rotated Griewank's Function (F_9) is a shifted, multi-modal, scalable and non-separable function. The function definition is given below (where $z = (x - o) * M$, M is a linear transformation matrix and $bias = -180$).

$$f(x) = \frac{1}{4000} \sum_{i=1}^{D} {z_i}^2 - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + bias, x \in [-600, 600]^D \tag{3.9}$$

Shifted Rotated Elliptic Function (F_{10}) is a shifted, uni-modal, scalable and non-separable function. The function definition is given below (where $z = (x - o) * M$, M is an orthogonal matrix and $bias = -450$).

$$f(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + bias, x \in [-100, 100]^D \tag{3.10}$$

Shifted Rotated Rastrigin's Function (F_{11}) is a shifted, multi-modal, scalable and non-separable function. This function also has many local minima, which makes it very difficult to find the global optimum value. The function definition is given below (where $z = (x - o) * M$, M is a linear transformation matrix and $bias = -330$).

$$f(x) = \sum_{i=1}^{D} ({z_i}^2 - 10\cos(2\pi z_i) + 10) + bias, x \in [-5, 5]^D \tag{3.11}$$

Shifted Rotated Ackley's Function (F_{12}) is a shifted, multi-modal, scalable and non-separable function. The function definition is given below (where $z = (x - o) * M$, M is a linear transformation matrix and $bias = -140$).

$$f(x) = -20exp\Bigg(- 0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {z_i}^2}\Bigg) - exp\Bigg(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi z_i)\Bigg) + 20 + e + bias, x \in [-32, 32]^D \tag{3.12}$$

All the above functions have known global minimum value, which is 0. In this study, for the functions $F_6 - F_{12}$, we used the same data set provided for CEC'2005 (Suganthan et al. 2005).

3.2 Standard Genetic Algorithms Test Problems

Again, many combinatorial optimization problems have been reported in literature to test the performance of evolutionary algorithms. In this study, we used the following combinatorial optimization problems, commonly known as standard genetic algorithms test problems (Reinelt 1995, Scholl & Klein 2003).

Traveling Salesman Problem (TSP) is an NP-hard combinatorial optimization problem. In this problem, a salesman has to visit *n* cities by starting the trip from a given city and returning back to the starting city to complete a *tour*. The constraint here is that each city should be visited exactly once. The purpose is to find the minimum length tour. We solved two instances of the problem: 29-cities TSP and 70-cities TSP. The distances between the cities were taken from the TSP library provided by Reinelt (1995); the best known solutions reported for the two problem instances are 2020 and 675, respectively (Reinelt 1995).

N-Queens Problem is also a classic NP-hard combinatorial optimization problem. In this problems, there are n queens and an n x n board. The purpose is to place all queens on the board such that no queen should *attack* the other one - a queen attacks another queen if they are in the same row, the same column or on the same diagonal. We solved three instances of the problem: 8-Queens, 16-Queens and 32-Queens.

Bin-Packing Problem is also proved to be an NP-hard combinatorial optimization problem. In this problem, there are n bins of varying capacities and m elements of varying sizes. The objective is to pack the bins with the maximum number of elements, without exceeding the maximum capacity of a bin. The exceeded value (to a bin's maximum capacity) is considered as an error. The total error, in all the bins, is computed as following (where C_i is the capacity and S_i is the total size of the elements in the i^{th} bin).

$$Error = \sum_{i:1}^{n}(|C_i - S_i|) \tag{3.13}$$

In this study, five instances of the benchmark data sets for the bin-packing problem (n1w1b1r0, n1w1b1r1, n1w1b1r2, n1w1b1r3, n1w1b1r4) provided by Scholl & Klein (2003) were used. In each data set, there are 50 bins with a total capacity 1000.

4 Empirical Study

Three primary aims were pursued in this study: a) system implementation, b) optimal parameter settings for EA/G and c) statistical analysis to compare the strength of EA/G against PBIL and CGA on the selected test problems. In the following, we discuss each of these in turn.

4.1 System Implementation

We used an empirical approach to compare the performance of EA/G, PBIL and CGA on a set of test problems. 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 below.

- The solutions were encoded as binary strings whose length varies from problem to problem as shown in Table 1.
- To keep uniformity, the probability vector was initialized to 0.5 for each competing algorithms.
- Initial population was sampled randomly; the population size was kept 100 throughout the study.

- Because, fitness of an individual is computed from its phenotype value rather than genotype, a function $binary2real$ was implemented which maps binary value (genotype) into corresponding decimal value (phenotype).
- In each generation, *best* and *average* fitness of population was recorded; for each algorithm total time elapsed was also recorded.
- Each algorithm terminated when the maximum number of generations exceeded a preset limit. The limit was set to 2.0e+05 generations for the numerical optimization functions and 1.2e+04 generations for the standard genetic algorithms test problems.
- Because, many combinatorial optimization problems place constraints on valid solutions, problem specific repair operators were built to transform an invalid solution to a valid solution. We adapted the template-based approach of Mitchell, Donoghue, Barnes & McCarville (2003), which, for example, corrects a tour (in TSP) by replacing the duplicate cities with the missing ones, taking the given template into account.

Table 1: Solution Encoding

Problem	**Encoding**	**Remarks**
F_1–F_{12}	300-bits string	10-consecutive bits for each dimension.
TSP	$n * log_2 n$-bits string	n is the number of cities; $log_2 n$ bits are interpreted as an integer value, which represents a (distinct) city in the tour.
N-queens	$n * log_2 n$-bits string	n is the number of queens; again, $log_2 n$ bits are interpreted as an integer value, which represents the column position of a queen on the board.
Bin-packing	$m * log_2 n$-bits string	n is the number of bins, and m is the number of items in each bin; each element to be packed is assigned a sequential substring of length $log_2 n$ whose value indicates the bin in which the element is placed.

4.2 Parameter Settings

Parameter tuning in an EA-based system is a challenging task and it needs a principled approach. In this study, we are investigating the optimal parameter values for EA/G only[3]. There are several parameters in EA/G, but to keep the parameter tuning process simple and manageable[4], we focused on *learning rate* (λ), *guided mutation rate* (β) and *population size* (N).

[3]We used the same parameter values for PBIL (*population size = 100*, *learning rate = 0.1*, *mutation probability = 0.02* and *mutation shift = 0.05*) as reported in Baluja (1994).

[4]We used the same parameter values for EA/G, for relatively less sensitive parameters (*mutation probability = 0.02*, *mutation shift = 0.05*, and *negative learning rate = 0.075*), as reported in Zhang et al. (2005).

To find the optimal values for the these parameters, we changed them in an orderly manner and recorded the performance of the algorithm as follows.

```
for $N$ = 10 : 10 : 100
  for $\lambda$ = 0.1 : 0.1 : 1.0
      for $\beta$ = 0.1 : 0.1 : 1.0
          Record fitness value
```

We performed 10 independent runs for this purpose to stabilize the parameter values for each test problem. We observed that a smaller value of λ (mostly 0.1) and a larger value of β (mostly 0.9) provided better results. We further fine tuned the parameter β; keeping $N = 100$ and $\lambda = 0.1$, it was observed that β = [0.95, 0.96, 0.97, 0.98] gave the best results. Therefore, in the rest of this study, we will be using $N = 100, \lambda = 0.1$ and $\beta = 0.95$ as optimal parameter values for EA/G.

4.3 Performance Test

The performance of an EA can be manifested in different ways. We are interested in two performance gains: a) *quality improvement* and b) *speed improvement*, but our main focus will be the solution quality. This is important, because even though most EAs use sophisticated strategies to find "good" solutions, finding an acceptably "good-enough" solution is not guaranteed, and 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 the quality criterion, if, keeping other things the same, the algorithm A attains a converged solution of higher fitness than the algorithm B. Similarly, an algorithm A beats an algorithm B under the speed criterion, if, keeping other things the same, the algorithm A attains a solution of a given quality/fitness in lesser time than the algorithm B. It is possible, however, that there could be a quality-speed trade-off, whereby 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 an algorithm B, we must show one of the following scenarios to be true:

P1 : Algorithm A performs better than the 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 Pentium-4 machine with 3.20 GHz speed and 1.50 GB of RAM. For each algorithms, we recorded the fitness of a solution for each generation, the best fitness by the end of a run, and the total time elapsed. All experiments were averaged over 20 independent runs.

5 Results

In this section, we present the simulation results regarding the performance of EA/G, PBIL and CGA on each test problem. To test whether the apparent differences in the performance gain are statistically significant, we also report on a one-tailed t-test.

5.1 Solution Quality

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 (Brest, Zamuda & Maučec 2010). Table 2 shows the simulation results of the three algorithms on each test problem in 20 independent runs, including the mean of best-of-run solution averaged over 20 runs (Mean), the standard deviation in best-of-run solution (σ), the overall best solution (Best), and the p value for a one-tailed t-test. The p value measures whether or not the pair-wise difference in the best solution for the competing algorithms (EA/G against PBIL, and EA/G against CGA) is statistically significant; in this study, $p < 0.05$ was our standard value for statistical significance.
It is evident from Table 2 that, overall, EA/G performed better than PBIL and CGA. The results also indicate that both EA/G and PBIL appear to offer competitive performance on simple optimization problems (F_1, F_2, F_3, F_4, F_6, F_7, 8-Queens, 16-Queens and 32-Queens). However, on relatively complex optimization problems (F_5, $F_8 - F_{12}$, all instances of Bin Packing and TSP), EA/G outperformed PBIL. Figure 1 plots the evolution of solutions for some selective problems. It was also observed that, on complex optimization problems, the average solution quality of EA/G at various points was greatly improved over PBIL and CGA. Pair-wise t-tests further revealed that the apparent differences are statistically significant. It is interesting to note that on separable functions, namely F_6 and F_7, both PBIL and EA/G offered competitive performance. However, for the non-separable functions ($F_8 - F_{12}$), the performance of EA/G was better than PBIL; explanation follows. It is also interesting to mention here that the performance of EA/G on functions $F_6 - F_{12}$ is also comparable to the results reported by Auger & Hansen (2005).

5.2 Speed

Here, we report on our second performance criterion, speed gain. We recorded the time (in milliseconds) taken by each algorithm to finish the search process. Again, all the simulation results were averaged over 20 independent runs. The results are shown in Table 3.
The results in Table 3 indicate that both PBIL and CGA converge more quickly than EA/G. Pairwise comparisons further revealed that the convergence time of CGA and PBIL is significantly faster than EA/G (see $p < 0.05$ in each case). These results suggest that EA/G is computationally more expensive than PBIL and CGA; explanation follows.

5.3 Discussion

The study presented here revealed two primary results. First, on complex optimization problems, EA/G outperformed both PBIL and CGA in attaining a good-quality solution. This indi-

Table 2: Solution fitness over a sample of 20 independent runs

	PBIL		EA/G		CGA		p-value†	
	Mean (σ)	**Best**	**Mean (σ)**	**Best**	**Mean (σ)**	**Best**	p_1	p_2
F_1	4.37e-06 (1.9e-08)	0.0	1.72e-06 (1.1e-08)	0.0	3.81e+02 (2.1e+01)	3.15e+02	–	< 0.01
F_2	2.13e-04 (8.0e-06)	0.0	1.37e-05 (7.4e-06)	0.0	7.12e+03 (7.9e+01)	5.63e+02	–	< 0.01
F_3	9.32e-01 (1.9e-01)	7.18e-02	4.71e-04 (2.7e-04)	2.85e-06	8.32e+02 (3.4e+01)	3.74e+02	0.47	< 0.01
F_4	3.61e-02 (2.3e-04)	5.82e-03	2.37e-03 (4.2e-04)	1.61e-03	3.24e+02 (5.8e+01)	2.83e+02	0.29	< 0.01
F_5	1.96e+01 (5.6e-06)	1.21e+01	4.39e-01 (3.6e-07)	**6.13e-02**	2.72e+03 (7.6e+01)	1.95e+03	< 0.01	< 0.01
F_6	7.62e-01 (9.3e-01)	2.31e-01	9.35e-01 (6.9e-04)	8.51e-03	6.8e+04 (2.9e+01)	5.12e+04	0.23	< 0.01
F_7	8.91e+01 (1.9e-01)	2.65e+01	6.42e+01 (1.5e-01)	1.47e+01	5.62e+03 (1.3e+01)	4.39e+02	< 0.01	< 0.01
F_8	5.73e+01 (7.9e-01)	3.27e+01	4.31e-02 (3.7e-01)	**2.18e-02**	7.46e+04 (2.7e+01)	7.12e+04	< 0.01	< 0.01
F_9	2.76e+00 (7.9e-01)	1.31e+00	1.95e+00 (3.7e-01)	**3.23e-02**	6.26e+02 (4.3e+01)	5.42e+02	< 0.01	< 0.01
F_{10}	2.86e+02 (4.8e-01)	1.52e+02	3.94e-02 (4.1e-01)	**5.73e-03**	9.71e+04 (1.7e+03)	9.14e+04	< 0.01	< 0.01
F_{11}	5.81e+03 (9.1e+01)	4.37e+03	8.12e+02 (2.3e+01)	**6.14e+02**	8.43e+05 (1.3e+03)	8.17e+05	< 0.01	< 0.01
F_{12}	5.86e+02 (6.3e+01)	3.95e+02	4.71e+01 (2.3e+01)	**3.82e+01**	8.34e+06 (1.3e+03)	8.21e+06	< 0.01	< 0.01
BP n1w1b1r0	3.47e+01 (4.9e+00)	2.82e+01	9.27e+00 (9.3e-01)	**6.83e+00**	8.73e+01 (1.7e+01)	8.36e+01	< 0.01	< 0.01
BP n1w1b1r1	4.71e+01 (7.1e+00)	4.37e+01	1.63e+01 (5.8e+00)	**1.48e+01**	9.73e+01 (1.3e+01)	9.25e+01	< 0.01	< 0.01
BP n1w1b1r2	6.92e+01 (7.1e+00)	6.17e+01	6.72e+01 (5.8e+00)	6.25e+01	3.47e+02 (1.3e+01)	2.72e+02	0.32	< 0.01
BP n1w1b1r3	8.13e+01 (5.9e+00)	7.65e+01	2.83e+01 (7.3e+00)	**2.51e+01**	1.73e+02 (6.1e+01)	1.39e+02	< 0.01	< 0.01
BP n1w1b1r4	3.85e+01 (4.5e+00)	3.32e+01	3.16e+01 (3.9e+00)	**2.71e+01**	8.52e+01 (1.6e+01)	7.96e+01	0.19	< 0.01
TSP70	3.2e+02 (4.3e+01)	2.67e+02	9.72e+01 (1.79e+01)	**6.38e+01**	1.37e+03 (1.9e+02)	9.93e+02	< 0.01	< 0.01
TSP29	8.76e+02 (6.7e+01)	6.72e+02	5.41e+02 (3.6e+01)	**4.27e+02**	7.51e+04 (1.7e+02)	7.26e+04	< 0.01	< 0.01
8 Queens	0 (0.0)	0	0 (0.0)	0	0 (0.0)	0	–	–
16 Queens	0 (0.0)	0	0 (0.0)	0	0 (0.0)	0	–	–
32 Queens	6.27e-01 (1.8e-02)	0	2.5e-01 (9.7e-03)	0	2.53e+00 (1.7e+00)	0	–	–

† p_1 and p_2 are the p values for a pair-wise test between PBIL and EA/G, and EA/G and CGA, respectively.

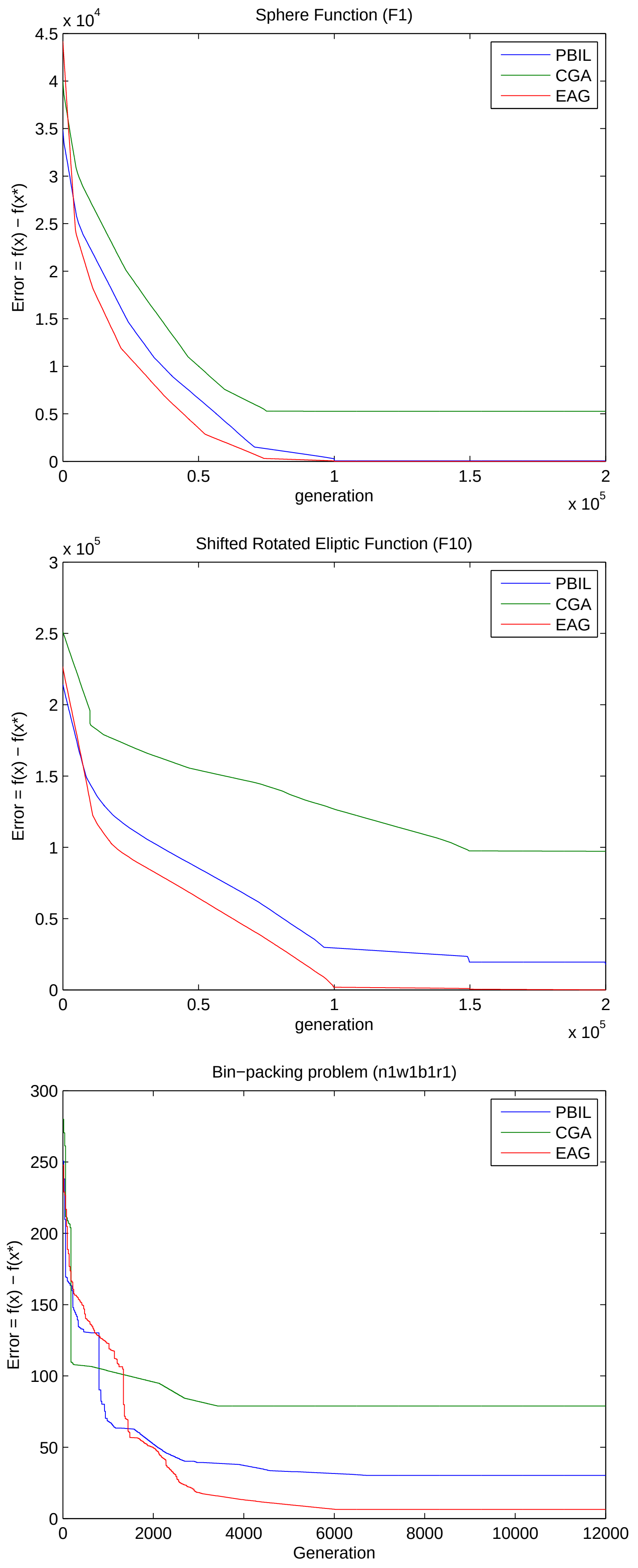

Figure 1: Convergence graph for F_1, F_{10} and Bin-packing problem (n1w1b1r1)

Table 3: Computational time† over a sample of 20 runs

	PBIL (σ)	**EA/G** (σ)	**CGA** (σ)	**p-value**‡	
				p_1	p_2
F_1	8.13e+05 (3.82e+01)	8.52e+05 (2.94e+01)	1.64e+03 (1.75+01)	0.25	< 0.01
F_2	8.39e+05 (3.84e+01)	1.32e+06 (6.14e+01)	2.36e+03 (1.29e+01)	< 0.01	< 0.01
F_3	9.64e+05 (1.27)	3.93e+06 (3.17e+01)	1.88e+03 (1.47e+01)	< 0.01	< 0.01
F_4	2.92e+06 (1.14e+01)	1.38e+07 (2.47e+01)	9.73e+03 (1.58e+01)	< 0.01	< 0.01
F_5	9.76e+05 (1.62e+01)	1.3e+06 (2.39e+01)	5.27e+03 (3.15e+01)	< 0.01	< 0.01
F_6	1.74e+06 (7.83e+01)	3.79e+07 (1.36e+02)	6.92e+04 (1.26e+01)	< 0.01	< 0.01
F_7	3.21e+07 (2.42e+01)	4.96e+07 (3.71e+01)	2.15e+05 (1.02e+01)	< 0.01	< 0.01
F_8	3.59e+08 (4.62e+01)	6.28e+08 (8.93e+01)	1.27e+07 (2.17e+01)	< 0.01	< 0.01
F_9	5.17e+07 (3.39e+01)	8.33e+07 (5.64e+01)	3.92e+05 (1.94e+01)	< 0.01	< 0.01
F_{10}	2.15e+10 (2.16e+02)	3.59e+10 (3.73e+02)	2.37e+08 (4.83e+01)	< 0.01	< 0.01
F_{11}	3.68e+08 (1.12e+02)	5.95e+08 (2.51e+02)	6.12e+06 (5.18e+01)	< 0.01	< 0.01
F_{12}	5.18e+06 (2.49e+02)	7.33e+08 (4.77e+02)	7.32e+04 (9.38e+01)	< 0.01	< 0.01
BP n1w1b1r0	3.25e+03 (1.66e+01)	6.17e+04 (2.13e+01)	1.91e+02 (1.18e+01)	< 0.01	< 0.01
BP n1w1b1r1	9.84e+03 (1.13e+01)	3.72e+04 (2.17e+01)	4.15e+02 (1.13e+01)	< 0.01	< 0.01
BP n1w1b1r2	6.73e+03 (2.17e+01)	9.18e+04 (3.52e+01)	2.16e+02 (1.91e+01)	< 0.01	< 0.01
BP n1w1b1r3	3.91e+03 (1.68e+01)	3.25e+04 (3.11e+01)	1.82e+02 (1.17e+01)	< 0.01	< 0.01
BP n1w1b1r4	5.18e+03 (1.19e+01)	7.17e+03 (1.85e+01)	2.91e+02 (1.02e+01)	< 0.01	< 0.01
TSP70	8.15e+04 (1.88e+01)	3.19e+05 (2.61e+01)	4.32e+02 (1.97e+01)	< 0.01	< 0.01
TSP29	2.37e+03 (1.32e+01)	3.86e+04 (3.72e+01)	2.19e+02 (2.38e+01)	< 0.01	< 0.01
8 Queens	2.34e+02 (3.18e+01)	5.12e+03 (7.63e+01)	1.17e+02 (2.37e+01)	< 0.01	< 0.01
16 Queens	7.18e+03 (1.96e+01)	6.92e+04 (5.27e+01)	1.25e+02 (2.43e+01)	< 0.01	< 0.01
32 Queens	9.27e+03 (1.73e+02)	1.14e+05 (6.81e+01)	7.12e+02 (5.12e+01)	< 0.01	< 0.01

†: Time was recorded in milliseconds.

‡ p_1 and p_2 are the p values for a pair-wise test between PBIL and EA/G, and EA/G and CGA, respectively.

cates that, in sampling offspring, a combination of global statistical information and the location information of the solutions found so far is better than using the global statistical information only. We observed that on separable functions, both PBIL and EA/G performed well but on non-separable functions, EA/G performed far better than PBIL. This suggests that the underlying assumption in EDAs that the problem variables are independent may prevent efficient convergence to the global optimum when problem variables interact strongly.
Second, both PBIL and CGA were found faster than EA/G. This suggests that EA/G is computationally expensive. The reason behind this expensiveness is that the *guided mutation operator* used in EA/G involves many computations in sampling offspring. 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 found more slowly, which of the two should be preferred? Perhaps solution quality should be given more weight as compared to speed.
Finally, there are rather high differences between the known global optimum solution and the solution found by the three algorithms. One explanation to these differences could be that our encoding scheme seems *suboptimal*. We encoded the solutions (genotype) as bit strings, 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, which as a whole degrades the performance of an algorithm in terms of speed as well.

6 Conclusion

This article described a comparative study of EA/G (Evolutionary Algorithm with Guided Mutation) against PBIL (Population-Based Incremental Learning) and CGA (Compact Genetic Algorithm). In a nutshell, we found that EA/G performed better than PBIL and CGA in attaining a good-quality solution. This confirms that a hybrid approach to use both global statistical information and the location information of the optimal solutions to sample offspring is better than using the the global statistical information only. On the other hand, both PBIL and CGA were found faster than EA/G. This confirms that EA/G is computationally expensive.
In future, we aim to analyze thoroughly the performance of PBIL and EA/G on large-scale global optimization problems (Tang, Li, Suganthan, Yang & Weise 2010) to see how scalable these algorithms are. We also intend to compare these algorithms with some recently developed algorithms to solve more application-oriented optimization problems (Guzmán, Gómez, Ardila & Jabba 2013, Purcaru, Precup, Iercan, Fedorovici, David & Dragan 2013, Torres, Quintana & Pinzón 2013). For example, Purcaru et al. (2013) used Gravitational Search Algorithm to find an optimal path for a robot in a complex environment. The present study also revealed that our solution encoding scheme is suboptimal, hence degrading the performance of an algorithm in attaining a good-quality solution. In the future work, we also aim to encode solutions as Edge Histogram Model (Tsutsui 2002). We conjecture that the Edge Histogram based sampling would greatly improve the performance of these algorithms.

Acknowledgment

We thank reviewers for their valuable comments how to improve the contents and presentation of this article.

References

Auger, A. & Hansen, N. 2005. Performance evaluation of an advanced local search evolutionary algorithm, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Edinburgh, UK, pp. 1777–1784.

Baluja, S. 1994. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning, *Technical Report CMU-CS-94-163*, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Baluja, S. & Davies, S. 1998. Fast probabilistic modeling for combinatorial optimization, *Proceedings of the* 15^{th} *National Conference on Artificial Intelligence (AAAI)*, pp. 469–476.

Brest, J., Zamuda, A. & Maučec, M. S. 2010. Large scale global optimization using self-adaptive differential evolution algorithm, *Proceedings of the IEEE World Congress on Computational Intelligence*, Barcelona, Spain, pp. 1–8.

Chen, S. H., Chang, P. C. & Zhang, Q. 2009. A self-guided genetic algorithm for flowshop scheduling problems, *Proceedings of the* 11^{th} *Conference on Evolutionary Computation*, IEEE Press, Piscataway, NJ, USA, pp. 471–478.

Guzmán, L. G., Gómez, A. S., Ardila, C. J. & Jabba, D. 2013. Adaptation of the GRASP algorithm to solve a multiobjective problem using the pareto concept, *International Journal of Artificial Intelligence* **11**(A13): 222–236.

Handa, H. 2005. Estimation of distribution algorithms with mutation, *Proceedings of the* 5^{th} *European Conference on Evolutionary Computation in Combinatorial Optimization*, Springer-Verlag, Berlin, Heidelberg, pp. 112–121.

Harik, C., Goldberg, D. E. & Miller 1997. The gambler's ruin problem, genetic algorithms, and the sizing of populations, *Proceedings of the* 4^{th} *International Conference on Evolutionary Computation*, New York: IEEE Press, pp. 7–12.

Harik, G. R., Lobo, F. G. & Goldberg, D. E. 1999. The compact genetic algorithm, *IEEE Transactions on Evolutionary Computation* **3**(4): 287–297.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, Cambridge, MA, USA.

Koza, J. R. & Poli, R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*, MIT Press, Cambridge, MA, USA.

Larrañaga, P. & Lozano, J. A. 2002. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, Boston, MA, USA.

Mitchell, G. G., Donoghue, D. O., Barnes, D. & McCarville, M. 2003. Generepair: A repair operator for genetic algorithms, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Chicago, USA, pp. 235–239.

Mühlenbein, H. & Paaß, G. 1996. From recombination of genes to the estimation of distributions I. binary parameters, *Proceedings of the 4^{th} International Conference on Parallel Problem Solving from Nature - PPSN IV*, Springer Berlin Heidelberg, pp. 178–187.

Niño, E. D. 2012. Samods and sagamods: Novel algorithms based on the automata theory for the multi-objective optimization of combinatorial problems, *International Journal of Artificial Intelligence (IJAI) – Special Issue on Metaheuristics in Artificial Intelligence* **8**(S12): 147–165.

Niño, E. D., Ardila, C., Donoso, Y. & Jabba, D. 2010. A novel algorithm based on deterministic finite automaton for solving the mono-objective symmetric traveling salesman problem, *International Journal of Artificial Intelligence* **5**(A10): 101–109.

Ochoa, A. & Soto, M. R. 2006. Linking entropy to estimation of distribution algorithms, *in* J. A. Lozano, P. Larrañaga, I. Inza & E. Bengoetxea (eds), *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, Springer, pp. 1–38.

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

Purcaru, C., Precup, R.-E., Iercan, D., Fedorovici, L.-O., David, R.-C. & Dragan, F. 2013. Optimal robot path planning using gravitational search algorithm, *International Journal of Artificial Intelligence* **10**(S13): 1–20.

Rechenberg, I. 1973. *Evolutionsstrategie: Optimierung Technischer Systeme und Prinzipien Der Biologischen Evolution*, Frommann-Holzboog, Stuttgart Germany.

Reinelt, G. 1995. Tsplib, universität heidelberg. Retrieved January 05, 2013, from http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

Santana, R., Larrañaga, P. & Lozano, J. A. 2008. Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem, *Journal of Heuristics* **14**(5): 519–547.

Scholl, A. & Klein, R. 2003. Bin-packing data-set 2 for BPP-1. Retrieved May 26, 2013, from http://www.wiwi.uni-jena.de/entscheidung/binpp/bin2dat.htm.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A. & Tiwari, S. 2005. Problem Definitions and Evaluation Criteria for the CEC-2005 Special Session on Real-Parameter Optimization, *Technical report*, Nanyang Technological University, Singapore.

Tang, K., Li, X., Suganthan, P. N., Yang, Z. & Weise, T. 2010. Benchmark functions for the CEC-2010 special session and competition on large scale global optimization, *Technical report*, Nature Inspired Computation and Applications Laboratory - USTC, China.

Torres, W. C., Quintana, M. & Pinzón, H. 2013. Differential diagnosis of hemorrhagic fevers using ARTMAP and an artificial immune system, *International Journal of Artificial Intelligence* **11**(A13): 150–169.

Tsutsui, S. 2002. Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram, *Proceedings of the* 7^{th} *International Conference on Parallel Problem Solving from Nature (PPSN-VII)*, Springer Berlin Heidelberg, pp. 224–233.

Yang, X.-S. 2010. Test problems in optimization, *in* X.-S. Yang (ed.), *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley & Sons.

Zhang, Q. & Mühlenbein, H. 2004. On the convergence of a class of estimation of distribution algorithms, *IEEE Transactions on Evolutionary Computation* **8**(2): 127–136.

Zhang, Q., Sun, J. & Tsang, E. 2005. An evolutionary algorithm with guided mutation for the maximum clique problem, *IEEE Transactions on Evolutionary Computation* **9**(2): 192–200.