Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 2565809, 10 pages
http://dx.doi.org/10.1155/2016/2565809

Research Article

A Guiding Evolutionary Algorithm with Greedy Strategy for Global Optimization Problems

Leilei Cao,[1,2] Lihong Xu,[1] and Erik D. Goodman[2]

[1]*Department of Control Science and Engineering, Tongji University, Shanghai 201804, China*
[2]*BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48824, USA*

Correspondence should be addressed to Lihong Xu; xulhk@163.com

Received 23 January 2016; Revised 15 April 2016; Accepted 3 May 2016

Academic Editor: Jens Christian Claussen

A Guiding Evolutionary Algorithm (GEA) with greedy strategy for global optimization problems is proposed. Inspired by Particle Swarm Optimization, the Genetic Algorithm, and the Bat Algorithm, the GEA was designed to retain some advantages of each method while avoiding some disadvantages. In contrast to the usual Genetic Algorithm, each individual in GEA is crossed with the current global best one instead of a randomly selected individual. The current best individual served as a guide to attract offspring to its region of genotype space. Mutation was added to offspring according to a dynamic mutation probability. To increase the capability of exploitation, a local search mechanism was applied to new individuals according to a dynamic probability of local search. Experimental results show that GEA outperformed the other three typical global optimization algorithms with which it was compared.

1. Introduction

Most real-world optimization problems in engineering, management, and other applications are of high dimensionality, multimodal, and/or multiobjective. Methods for solving these problems range from classical mathematical methods to heuristic algorithms. With more complex problems, heuristic intelligent algorithms have been used successfully and frequently [1]. These algorithms include Simulated Annealing [2], Genetic Algorithm [3], Particle Swarm Optimization [4], Ant Colony Optimization [5], Artificial Bee Colony [6], Firefly Algorithm [7, 8], and Bat Algorithm [9]. Support for parallel computation and stochastic search are common characteristics of these algorithms. Inevitably, most of these algorithms are slow to converge or converge prematurely, trapped in local optima upon addressing high-dimensional or multimodal optimization problems [10]. High-dimensional variables expand the search space, influencing the required number of evaluations and speed of convergence. Multimodality means that there exist one or more global optima and some number of locally optimal values [10, 11]. The "No Free Lunch" theorem clarifies that the performance of any search algorithm depends on the complexity of the problem domain [12].

All of the above-mentioned heuristic algorithms are population-based except Simulated Annealing, and some of them are classified as swarm intelligence. These population-based algorithms, when applied to complex problems, frequently outperform classical methods such as linear programming [13], gradient methods [14], and greedy algorithms [15]. Although the heuristic algorithms can solve some problems, sometimes they converge on local optima or find solutions near but not at the global optimum.

This paper introduces a Guiding Evolutionary Algorithm to address global optimization problems. It is inspired by Particle Swarm Optimization and the Bat Algorithm and uses some methods from the Genetic Algorithm. It is easily understood and implemented. For some problem classes tested, this algorithm has outperformed canonical implementation of those individual heuristic methods. This paper is organized as follows: Section 2 gives brief overview about PSO and BA. Section 3 proposes this new global optimization

algorithm. A set of test functions and the experimental results are presented in Section 4, and conclusions are presented in Section 5.

2. Particle Swarm Optimization and Bat Algorithm

2.1. Overview of PSO. A variety of heuristic algorithms are called swarm intelligence algorithms, simulating collaborative behavior of a swarm of insects, birds, fish, or other animals searching for food [4, 16]. Among them, Particle Swarm Optimization (PSO) is the most widely used. PSO was first proposed by Eberhart and Kennedy [4] in 1995. Each solution in PSO is a "particle" that searches in the independent variable space by learning from its own experience and other particles' past experiences [17, 18]. There are two different ways of representing others' past experiences: the global best and the local best. In global optimization problems, each particle adjusts its velocity and position according to the historical best position of itself and the best position among all particles. But this method tends to make each particle converge to a local optimum in those problems containing many locally optimal values. For these problems, the local best method is more effective. Each particle adjusts its velocity and position according to its historical best position and the best solution found within its neighborhood. In fact, if the neighborhood is all particles, the local best becomes the global best version. PSO is simple and easy to be implemented, but PSO lacks a mutation mechanism which could increase the diversity of the population and help to avoid convergence at local optima.

2.2. Bat Algorithm [9, 19–21]. The Bat Algorithm (BA) was proposed by Yang in 2010. It is based on the echolocation behavior of bats, which can find their prey and discriminate different types of insects even in complete darkness. This algorithm combines global search with local search in each iteration, which balances exploration and exploitation during search. The algorithm defines the rules for how these bats' positions and velocities in a d-dimensional search space are updated. The new positions and velocities at time step t are given by

$$\begin{aligned} f_i &= f_{\min} + \left(f_{\max} - f_{\min}\right) * \beta, \\ v_i^t &= v_i^{t-1} + \left(x_i^{t-1} - x_*^{t-1}\right) * f_i, \\ x_i^t &= x_i^{t-1} + v_i^t, \end{aligned} \tag{1}$$

where $\beta \in [0, 1]$ is a random vector and x_*^{t-1} is the global best location found to date. f_i is the frequency of the wave.

For the local search part, a new solution for each bat is generated locally using a random walk around the current best solution:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon * A^t, \tag{2}$$

where $\varepsilon \in [-1, 1]$ is a uniformly distributed random number and A^t is the average loudness of all bats at this time step.

The update of the velocities and positions of bats has some similarity to the procedure in the standard Particle Swarm Optimization as f_i essentially controls the pace and range of movement of the swarming particles. To a degree, BA can be considered as a balanced combination of the standard Particle Swarm Optimization and an intensive local search controlled by the loudness and pulse rate. Furthermore, the loudness A_i and rate r_i of pulse emission must be updated accordingly as the iterations proceed. The loudness usually decreases once a bat has found its prey, while the rate increases.

3. A Guiding Evolutionary Algorithm

Although Bat Algorithm can solve some difficult problems and converge quickly, it often cannot avoid converging to a local optimum, especially for those high-dimensional multimodal problems. Examining PSO and BA reveals some differences, in that BA rejects the historical experience of each individual's own position but accepts a better personal solution with some probability. We will modify some of the updating mechanisms of BA and add a mutation method in order to try to address global optimization problems more accurately.

3.1. Crossover. There is a leading individual in both PSO and BA, which can lead all individuals to update their own positions. But we find that current velocities of individuals are weighted accumulations of their own historical velocities across iterations. In the later parts of a search, this will tend to decrease the speed of convergence or favor convergence on local optima, even though there are bounds on velocities in PSO. To avoid these situations and employ the leading individual effectively, we use the following expression:

$$x_i^t = x_i^{t-1} + \left(x_*^{t-1} - x_i^{t-1}\right) * \beta, \tag{3}$$

where x_*^{t-1} is the current global best individual and β is the step length of the position increment.

In fact, this expression is a form of crossover between each individual and the best individual to generate their offspring. x_*^{t-1} is a guide to lead all other individuals to walk forward toward the current best one. To explain formula (3) clearly, we use a simple example to describe the process of locating the global best individual. Figure 1 shows a two peaks' function, the ordinate of which is the value of the objective function and the abscissa of which is the design variable. Point P is the global best we expect to find in this maximization problem. Points A, B, C, and D indicate four individuals, and x_1, x_2, x_3, and x_4 are their solutions in variable space, respectively. $x_{\min}$ and $x_{\max}$ are the bounds of the variable space. Among the four individuals, point C is the best; therefore, other individuals will move toward this current best individual. Let us assume $\beta \in [0, 2]$ is a uniform random variable. For point A, its motion ranges between x_1 and $x_{\max}$, because $2 * (x_3 - x_1)$ exceeds the upper bound of the variable range. Theoretically, point A can move to each point whose variable value is between x_1 and $x_{\max}$, and this will depend on the random variable β. Not only point A but also points B and D will be attracted by point C. The best individual will be replaced once a better solution occurs. After many iterations, all of these individuals can reach the global best point P. While searching,

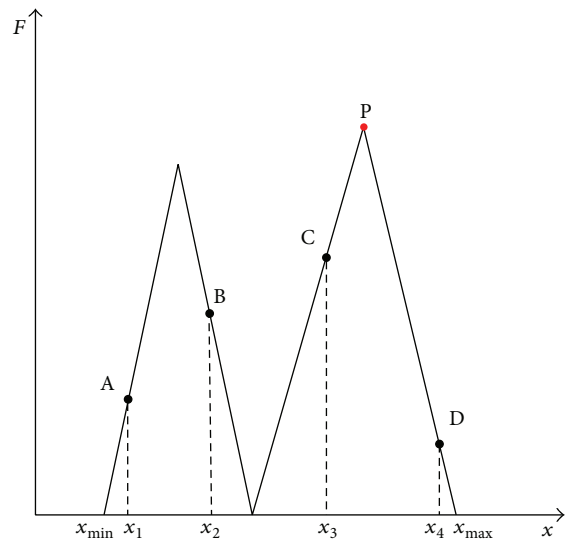

Figure 1: A two peaks' function.

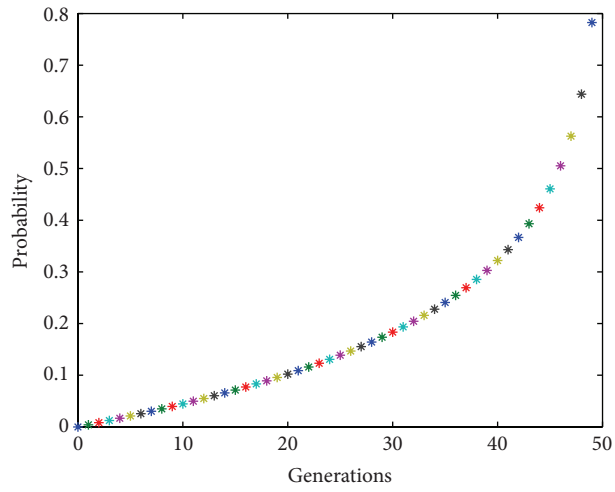

Figure 2: Probability of mutation.

even though the best individual may reach the lower peak, other individuals can also move to point P because of random walking and a great number of population.

3.2. Mutation. Although these global optimization algorithms work quite well on some engineering problems, they sometimes cannot avoid converging on a local optimum with high probability. For such heuristic algorithms, it is difficult to prove that they will always converge to a global optimum. In order to raise the probability of converging to a global optimum, we add a mutation mechanism which is inspired by the Genetic Algorithm. The purpose of mutation is to increase the diversity of the population and prevent them trapping into a local optimum, especially in the later iterations. Therefore, the probability of mutation will be made low at the beginning and higher later. We set the mutation probability as follows:

$$p = c * \ln\left(\frac{T_{\max}}{T_{\max} - t}\right), \tag{4}$$

where c is a limiting parameter which can be a constant or a variable, $T_{\max}$ is the maximum number of generations, and t is the current generation. We will assume that $T_{\max} = 50$ and $c = 0.2$; then, probability p increases with iterations as shown in Figure 2. Especially in the early iterations, p is low, while it increases rapidly in the later iterations.

The mutation formula is as follows:

$$x_i^t = x_i^t + \varepsilon * M, \tag{5}$$

where x_i^t is the solution of an individual after crossover, $\varepsilon \in [-1, 1]$ is a uniform random number, and M is a vector which determines the scope of mutation. In general, we set M such that mutation can span the entire range of the independent variables. We will assume that the range of jth dimension of the variables is $[a, b]$, and then

$$M_j = \max\left(x_{ij}^t - a, b - x_{ij}^t\right). \tag{6}$$

Each dimension of the vector M is set as formula (6). Sometimes the solution after mutation will exceed the range of the variables, and we can limit it to the bounds in this situation.

3.3. Local Search. As we know, most of global optimization algorithms have excellent capability of exploration but are weak at exploitation. To enhance this ability, especially in the later iterations, we will expect the algorithm to be able to locate the global best quickly with local search, once it has found the right neighborhood. The probability of local search will be maintained low in early iterations and raised later in the search process. The probability of local search will follow the same distribution as mutation (4).

Each individual has the calculated probability of being replaced by a result of local search, performed around the current best individual. Similarly to the Bat Algorithm's random walk, we use the following formula to express local search:

$$x_i^t = x_*^{t-1} + \varepsilon * L, \tag{7}$$

where x_*^{t-1} is the best individual of the current population, $\varepsilon \in [-1, 1]$ is a uniform random number, and L is a vector which determines the search scope of the random walk, formulated in the variable space. We still will assume that the range of jth dimension of the variables is $[a, b]$, and then

$$L_j = 10\% * (b - a). \tag{8}$$

10% is only an example of scope of the random walk. The other dimensions of the vector L are set as formula (8). Local search acts much like mutation; the differences are in the points to which the mutational changes are added—the global best in one case and any individual in the other.

3.4. Summary of GEA. Unlike the classical (greedy) simplex algorithm, a population-based algorithm can "jump out" of a local optimum using parallel search, without losing all prior

Initialize the population x_i $(i = 1, 2, \ldots, n)$, and define limit parameter c,
scope of mutation M, range of local search L
Evaluate the initialized population
Select the best individual x_*^{t-1}
While $(t < T_{\max})$ {
For each individual {
Make crossover to generate a new individual x_i^t
If (rand < p) {
Make mutation for x_i^t
}
If (rand < p) {
Make local search for x_i^t
}
If ($f(x_i^t)$ is better than $f(x_i^{t-1})$) {
Accept this new individual
}
If ($f(x_i^t)$ is better than $f(x_*^{t-1})$) {
Replace x_*^{t-1} using x_i^t
}
}
}
Output results and visualization

Algorithm 1

information. Therefore, we can afford to accept a new global best-to-date individual when it is found, instead of accepting it according to some probability, as Simulated Annealing must do in order to reduce getting trapped at local optima. In the GEA, each individual updates only when its offspring has better fitness. The updates of the global best individual and each individual are greedy strategies, but, because of a large population size, convergence to local optima can be avoided in some multimodal global optimization problems.

The proposed Guiding Evolutionary Algorithm with greedy strategy resembles the framework of the Genetic Algorithm, which can be described as follows: initialization, evaluation, selection, crossover, and mutation. A difference from the Genetic Algorithm is that there is no special selection mechanism in the GEA, because each individual will generate its offspring by recombination with the global best individual and it does not require an operator to select an individual to evolve. In addition, local search is also employed to increase the algorithm's exploitation capability. The whole framework is easy to understand and implement. The whole pseudocode of this algorithm is as shown in Algorithm 1.

4. Experimental Results and Discussion

4.1. Test Functions [11, 22, 23]. In order to test the proposed algorithm, 6 commonly used functions are chosen. F_1–F_6 are well-known test functions for global optimization algorithms including Genetic Algorithm, Particle Swarm Optimization, Bat Algorithm, Firework Algorithm [24], and Flower Pollination Algorithm [25]. F_1 is a simple unimodal and convex function; F_2 is a unimodal but nonconvex function; F_3 is multimodal, since the number of local optima increases with the dimensionality; F_4 has a global minimum at 0, which is inside the parabolic, narrow-shaped flat valley (variables are strongly dependent on each other; therefore it is difficult to converge on the global optimum); F_5 is multimodal and has many local minima but only one global minimum; F_6 also has numerous local minima and is multimodal. The definitions of these benchmark functions are listed in Table 1. All of their global minima are at 0.

4.2. Parameter Settings for Testing Algorithms. In this section, the Guiding Evolutionary Algorithm (GEA), Fast Evolutionary Algorithm (FEA) [23], Particle Swarm Optimization (PSO), and Bat Algorithm (BA) are tested separately on six benchmark functions. For F_1 to F_6, to illustrate how the dimensionality of problems influences the performance of the algorithms, three different dimensions of benchmark functions are used: $D = 10$, $D = 20$, and $D = 30$. The parameters of the Fast Evolutionary Algorithm are set as in [23]. Through some trials, we set the number of generations in FEA, PSO, and BA as 500, while the number of generations is set as 50 in GEA, in order to get the best results. Population sizes were adjusted according to the different benchmark functions and different dimensions. Although these algorithms run for different numbers of generations, the numbers of evaluations are the same for any given benchmark function. For PSO, according to some researches [17, 18], the following settings can make it perform well: the inertia weight is set as 0.7, while the two learning parameters are set as 1.5; the limitation of velocity is set as $[-1, 1]$. Parameters of BA are set as follows: minimum frequency $f_{\min} = 0$, maximum frequency $f_{\max} = 2.0$, the loudness $A_0 = 0.8$, the pulse rate $r_0 = 0.8$, and the control factor $\alpha = \beta = 0.9$. These parameters are attributed to some published articles [11, 19]. For our GEA, $\beta \in [0, 2]$ and $c = 0.2$.

Table 1: Six test functions utilized in this experiment.

Functions	Name of function	Expression	Domain of variables
F_1	De Jong's sphere function	$f(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]$
F_2	Schwefel 2.22 function	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-15, 15]$
F_3	Griewangk's function	$f(x) = -\prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \sum_{i=1}^{D} \frac{x_i^2}{4000} + 1$	$[-15, 15]$
F_4	Rosenbrock's function	$f(x) = \sum_{i=1}^{D-1} 100 * \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$	$[-15, 15]$
F_5	Rastrigin's function	$f(x) = D * 10 + \sum_{i=1}^{D} \left(x_i^2 - 10 * \cos\left(2\pi x_i\right)\right)$	$[-5, 5]$
F_6	Ackley's function	$f(x) = 20 + e - 20 * \exp\left(-0.2 * \sqrt{\frac{1}{D} * \sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D} * \sum_{i=1}^{D} \cos\left(2\pi x_i^2\right)\right)$	$[-15, 15]$

Generally speaking, for unimodal functions, the convergence rate is of primary interest, as optimizing such functions to a satisfactory accuracy is not a major issue. For multimodal functions, the quality of final solutions is more significant since they can illustrate the ability of the algorithm tested, to escape from local optima and locate the true global optimum. Based on these requirements, we present the computational optimal solutions of these benchmarks using our GEA as well as comparing with other algorithms. In addition, the curves of convergence rate are plotted in order to indicate the numbers of evaluations to reach the approximate global optimum or converge on a local optimum.

4.3. Experimental Results and Discussion

4.3.1. Results for the 10D Benchmark Functions. Table 2 depicts the computational results of four algorithms on six benchmark functions with 10 dimensions, which were run 20 times independently to get the best, worst, mean, median, and standard deviation values. Among these five values, the "median" item can indicate the algorithm's optimizing performance reasonably, while the "standard deviation" item denotes the stability of the algorithm's optimizing performance. Figure 3 presents the convergence characteristics in terms of median value of 20 runs of each algorithm on each benchmark function.

From the computational results, we find that BA performs best among these four algorithms on F_1, although three of them have excellent performances. As stated earlier, the convergence rate is more crucial in comparison with convergence accuracy for unimodal functions. As indicated from Figure 3(a), GEA, PSO, and BA have fast convergence rate and high convergence accuracy, but FEA gets the worst accuracy. Actually, F_1 is only a simple unimodal and convex function. There is no mutation operation in the BA's procedure but only local search. However, compared with GEA and PSO, the local search and absence of mutation become the BA's advantage in addressing unimodal and convex functions. But this advantage is lose upon testing unimodal and nonconvex function F_2. GEA clearly outperforms the other algorithms in convergence rate and convergence accuracy; the median and mean value are close to the true global optimum. Viewed from Figure 3(b), it is easy to see that GEA and PSO have similar search processes in the early evaluations, but GEA can jump out from a local optimum because of its mutation operation and continue to search with local search, while PSO stops at the local optimum. GEA improves several orders of magnitude compared to the results of the other algorithms. In addition, it has better stability of optimization performance. F_3 is a multimodal function; both GEA and PSO perform well, and they have similar convergence rates and accuracies as indicated from Figure 3(c). We display the ability to avoid falling into local optima. As described, function F_4 poses special difficulties for an optimizer. Although PSO locates the minimization value exactly in median value on F_4, PSO's stability is worse than that of GEA. Sometimes, locating the exact optimum is not our purpose, but rather the stable computational performance of the algorithm is more significant. Each of GEA's values on F_4 is close to the real global optimum, and the stability of its optimizing performance is excellent. This performance benefits from changeable mutation parameters and local search areas. But, as viewed from Figure 3(d), the convergence accuracies of FEA and BA are worse than those of GEA and PSO. F_5 and F_6 are multimodal functions, but F_5 makes it extremely difficult to locate the global optimum, while F_6 is much easier. GEA and PSO have similar performance upon addressing F_5 and F_6, while GEA is a little better than PSO especially in standard deviation. Although neither of them can locate the real global minimization on F_5, they still work better than FEA and BA. As indicated from Figure 3(e), although FEA has the fastest convergence rate, the convergence accuracy is too poor similar to that of BA. In addition, BA also performs well upon addressing F_6, but it is not yet better than GEA and PSO. Their

Table 2: Results of benchmark functions in 10 dimensions using four algorithms.

Functions	Evaluations	Value	GEA	FEA	PSO	BA
F_1	50000	Best	$3.45E-37$	$2.31E-15$	$5.42E-38$	**$1.20E-44$**
		Worst	$6.62E-35$	$1.95E-03$	$1.05E-34$	**$5.57E-44$**
		Mean	$5.86E-36$	$4.83E-04$	$1.34E-35$	**$2.41E-44$**
		Median	$1.44E-36$	$2.89E-04$	$1.48E-36$	**$2.37E-44$**
		StDev	$1.46E-35$	$5.98E-04$	$2.88E-35$	**$1.05E-44$**
F_2	50000	Best	$5.25E-11$	$1.23E+00$	$3.79E-04$	**$3.74E-22$**
		Worst	**$3.44E-04$**	$7.27E+00$	$2.11E-01$	$5.11E+01$
		Mean	**$1.28E-04$**	$3.89E+00$	$6.86E-02$	$7.84E+00$
		Median	**$5.03E-05$**	$3.28E+00$	$2.97E-02$	$1.23E+00$
		StDev	**$1.20E-04$**	$1.96E+00$	$8.52E-02$	$1.42E+01$
F_3	50000	Best	$2.46E-02$	$2.34E-02$	**$7.40E-03$**	$2.46E-02$
		Worst	**$1.06E-01$**	$3.37E-01$	$1.62E-01$	$2.63E-01$
		Mean	$6.37E-02$	$1.36E-01$	**$4.88E-02$**	$1.24E-01$
		Median	$6.52E-02$	$1.31E-01$	**$3.69E-02$**	$1.08E-01$
		StDev	**$2.39E-02$**	$7.12E-02$	$3.82E-02$	$6.86E-02$
F_4	200000	Best	$4.23E-29$	$4.60E-03$	**$0E+00$**	$1.78E+00$
		Worst	**$6.91E-10$**	$6.16E+00$	$3.99E+00$	$6.98E+00$
		Mean	**$3.52E-11$**	$2.99E+00$	$5.98E-01$	$4.44E+00$
		Median	$3.88E-21$	$2.04E+00$	**$0E+00$**	$4.43E+00$
		StDev	**$1.54E-10$**	$2.55E+00$	$1.46E+00$	$1.31E+00$
F_5	200000	Best	$4.97E+00$	$6.96E+00$	**$3.98E+00$**	$6.96E+00$
		Worst	**$1.19E+01$**	$2.69E+01$	$1.39E+01$	$2.69E+01$
		Mean	**$7.56E+00$**	$1.71E+01$	$7.71E+00$	$1.77E+01$
		Median	**$7.46E+00$**	$1.59E+01$	$7.46E+00$	$1.69E+01$
		StDev	**$1.69E+00$**	$5.82E+00$	$2.77E+00$	$4.82E+00$
F_6	100000	Best	**$7.99E-15$**	$1.16E+00$	$7.99E-15$	$7.99E-15$
		Worst	**$1.65E+00$**	$3.40E+00$	$1.65E+00$	$2.32E+00$
		Mean	**$1.40E-01$**	$2.11E+00$	$3.96E-01$	$7.13E-01$
		Median	**$7.99E-15$**	$2.17E+00$	$7.99E-15$	$7.99E-15$
		StDev	**$4.38E-01$**	$7.93E-01$	$6.34E-01$	$9.39E-01$

median values are very close to the real global minimum. The convergence rates of GEA, PSO, and BA are also similar, but FEA has been trapped in a local optimum prematurely as seen from Figure 3(f).

4.3.2. Results for the 20D and 30D Benchmark Functions. Table 3 shows these four algorithms' computational results on six 20D benchmark functions, and Table 4 shows computational results on six 30D benchmark functions. As the convergence graphs are similar to those of the 10D functions except the convergence accuracy, they are not presented here. It is easy to see that increasing the dimension of F_1 did not influence the nature of the performance of these algorithms, just increasing the number of evaluations needed to get similar accuracy. Increased dimension of F_2 reduces the convergence accuracy of GEA, PSO, and BA, but FEA is not affected by that because FEA can never get good accuracy. Sometimes we can accept the results of GEA and PSO on F_2 in 20D and 30D, but that of FEA and BA cannot be accepted. The effects of F_3 on 20D and 30D are better than those on 10D for these four algorithms. This problem is known to become easier as the dimension increases [26]. GEA performs best among the methods on F_3, although increasing the problem's dimension and its convergence accuracy brings the computational optimum closer to the real global optimum for all tested methods. For F_4, F_5, and F_6, with increasing dimensions, none of these algorithms can locate the real global optimum or approach it closely. But GEA is the best compared with the other algorithms in optimization results. High dimension is a calamity for these problems, which makes it difficult to locate their real global optimum.

4.3.3. Discussion. Analysis of the results of four algorithms on six functions with different dimensions indicates that GEA has better optimization performance compared to FEA, PSO, and BA. BA is suitable to address simple unimodal problems, but it has an undistinguished performance on other problems, especially on multimodal problems. FEA has a fast convergence rate, but its convergence accuracy is too bad in some problems. Sometimes GEA and PSO have similar computational results and convergence rate and accuracy on multimodal problems, but GEA performs more

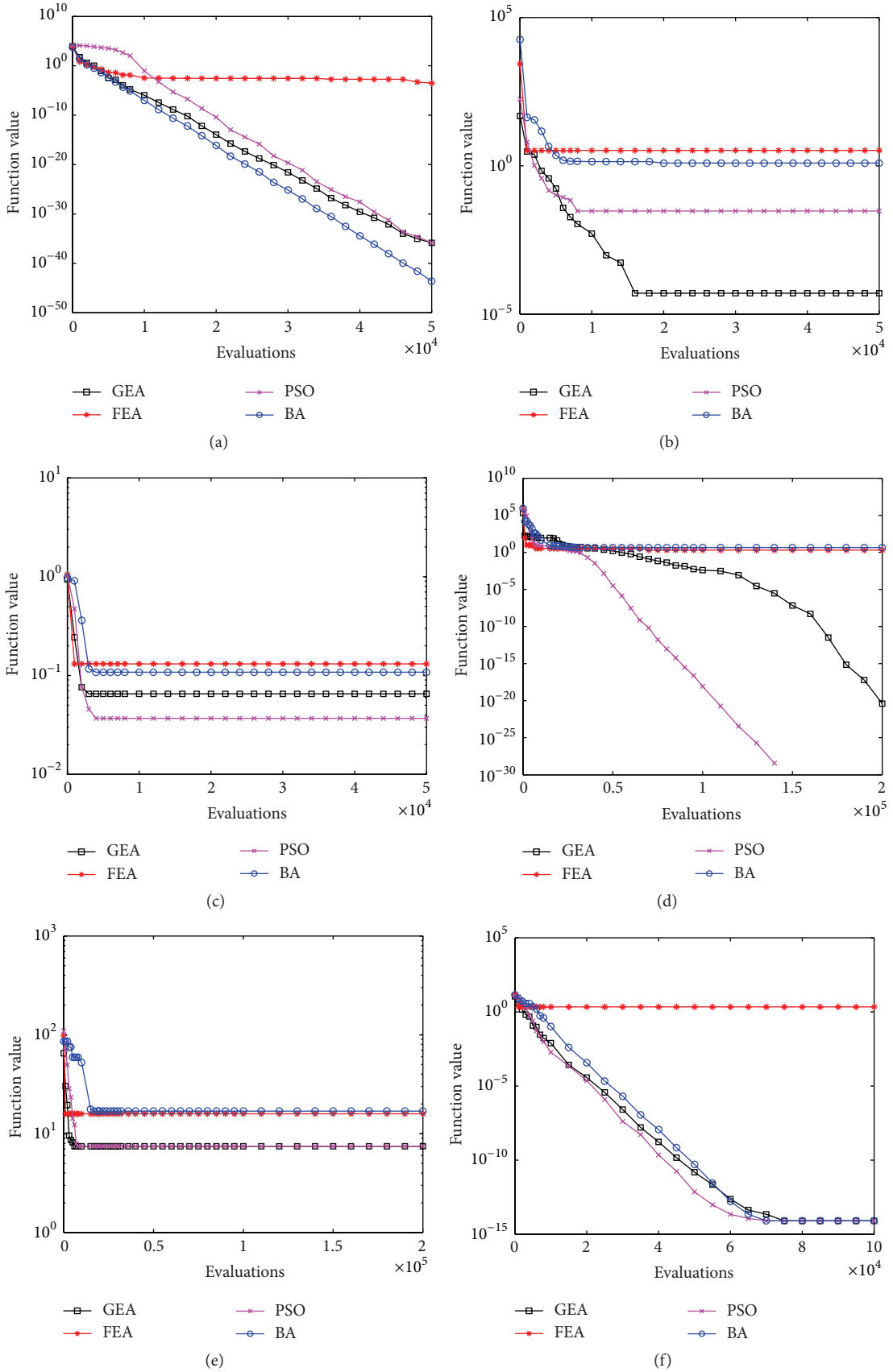

Figure 3: The median value convergence characteristics of 10D benchmark functions. (a) F_1; (b) F_2; (c) F_3; (d) F_4; (e) F_5; and (f) F_6.

Table 3: Results of benchmark functions in 20 dimensions using four algorithms.

Functions	Evaluations	Value	GEA	FEA	PSO	BA
F_1	200000	Best	$1.56E-36$	$2.26E-09$	$4.59E-30$	**4.07E − 44**
		Worst	$4.72E-35$	$6.20E-04$	$5.30E-15$	**1.18E − 43**
		Mean	$1.30E-35$	$1.84E-04$	$3.56E-16$	**7.51E − 44**
		Median	$7.39E-36$	$9.57E-05$	$4.45E-26$	**7.40E − 44**
		StDev	$1.37E-35$	$2.02E-04$	$1.22E-15$	**2.06E − 44**
F_2	200000	Best	$4.23E-03$	$1.24E+00$	$3.57E-03$	**8.71E − 22**
		Worst	**3.07E − 01**	$4.32E+00$	$1.36E+00$	$1.31E+02$
		Mean	**6.83E − 02**	$3.11E+00$	$2.29E-01$	$6.54E+01$
		Median	**5.30E − 02**	$3.13E+00$	$1.10E-01$	$7.71E+01$
		StDev	**6.86E − 02**	$9.24E-01$	$3.32E-01$	$4.39E+01$
F_3	200000	Best	**0E + 00**	$1.95E-03$	$0E+00$	$1.11E-16$
		Worst	**7.40E − 03**	$8.13E-02$	$1.23E-02$	$7.13E-02$
		Mean	**1.85E − 03**	$2.74E-02$	$2.22E-03$	$2.93E-02$
		Median	**1.11E − 16**	$2.30E-02$	$1.67E-16$	$2.22E-02$
		StDev	**3.29E − 03**	$1.92E-02$	$4.07E-03$	$2.19E-02$
F_4	400000	Best	**3.20E + 00**	$2.21E-02$	$5.33E+00$	$1.39E+01$
		Worst	**1.26E + 01**	$4.93E+01$	$1.69E+01$	$1.51E+02$
		Mean	**8.90E + 00**	$1.80E+01$	$1.12E+01$	$3.92E+01$
		Median	**9.34E + 00**	$1.90E+01$	$1.16E+01$	$1.59E+01$
		StDev	**2.00E + 00**	$9.30E+00$	$2.56E+00$	$4.58E+01$
F_5	400000	Best	**7.96E + 00**	$1.29E+01$	$7.96E+00$	$1.68E+02$
		Worst	**2.39E + 01**	$8.56E+01$	$2.89E+01$	$2.35E+02$
		Mean	$1.79E+01$	$4.55E+01$	**1.66E + 01**	$1.93E+02$
		Median	$1.89E+01$	$4.58E+01$	**1.64E + 01**	$1.92E+02$
		StDev	**3.73E + 00**	$2.04E+01$	$6.29E+00$	$1.67E+01$
F_6	200000	Best	**1.51E − 14**	$1.18E+00$	$1.51E-14$	$1.33E+01$
		Worst	**2.17E + 00**	$3.95E+00$	$2.45E+00$	$1.54E+01$
		Mean	**8.92E − 01**	$2.95E+00$	$9.60E-01$	$1.44E+01$
		Median	**1.16E + 00**	$3.04E+00$	$1.16E+00$	$1.45E+01$
		StDev	$7.86E-01$	**6.62E − 01**	$7.89E-01$	$6.77E-01$

stably because of its changeable mutation operation and local search, which can increase its capability of exploration and exploitation. The changeable mutation operation will reduce the convergence rate, but the greedy strategy offsets this reduction. Therefore, GEA has a convergence rate as quick as that of PSO and BA, and the convergence accuracy of GEA is better than/or similar to that of the others.

5. Conclusion

In this paper, a Guiding Evolutionary Algorithm (GEA) has been proposed to solve global optimization problems. This algorithm mixes advantage of Particle Swarm Optimization, Bat Algorithm, and Genetic Algorithm. The largest difference is that the GEA accepts a newly generated individual only when its fitness is better than that of the parent, while the PSO and many versions of the GA always accept the new individual. The BA accepts the better solution according to some given probability. This mechanism of GEA can guarantee that each individual stays in its historically best position and moves forward toward the global best position found to date. Comparing with PSO and BA, the velocity was removed from GEA but replaced by a random walk step toward the current global best individual. After crossover with the global best individual, mutation was added to the generated individual according to a mutation probability. To increase the algorithm's exploitation power, a local search mechanism was applied to a new individual according to a given probability of local search; this mechanism actually produces a random walk around the current global best individual.

Experimental results show that GEA can approximate the globally optimal solution very accurately in most of the problems tested. Comparing with three other typical global optimization algorithms, GEA performs more accurately and stably. In the future work, niching methods will be considered for addition to the GEA to solve those multimodal problems yet more effectively.

Table 4: Results of benchmark functions in 30 dimensions using four algorithms.

Functions	Evaluations	Value	GEA	FEA	PSO	BA
F_1	400000	Best	$9.32E-35$	$1.22E-08$	$6.14E-12$	**$1.06E-43$**
		Worst	$6.01E-33$	$3.86E-04$	$2.15E-07$	**$2.42E-43$**
		Mean	$1.28E-33$	$5.09E-05$	$1.99E-08$	**$1.62E-43$**
		Median	$6.57E-34$	$1.86E-05$	$8.45E-10$	**$1.62E-43$**
		StDev	$1.467E-33$	$8.90E-05$	$5.01E-08$	**$3.41E-44$**
F_2	400000	Best	$2.18E-02$	$1.73E+00$	$7.29E-01$	**$9.46E-22$**
		Worst	**$6.98E-01$**	$5.65E+00$	$9.51E-01$	$1.93E+02$
		Mean	**$2.66E-01$**	$3.37E+00$	$3.90E-01$	$7.45E+01$
		Median	**$2.35E-01$**	$3.30E+00$	$3.21E-01$	$6.32E+01$
		StDev	**$1.89E-01$**	$1.11E+00$	$2.45E-01$	$7.04E+01$
F_3	400000	Best	**$8.88E-15$**	$2.76E-02$	$4.68E-12$	$1.11E-16$
		Worst	**$9.86E-03$**	$1.43E-01$	$9.88E-03$	$2.71E-02$
		Mean	**$1.36E-03$**	$7.59E-02$	$2.10E-03$	$8.99E-03$
		Median	**$2.98E-13$**	$6.67E-02$	$6.39E-06$	$8.63E-03$
		StDev	**$3.34E-03$**	$3.53E-02$	$3.77E-03$	$7.79E-03$
F_4	600000	Best	**$1.37E+01$**	$2.90E+00$	$2.01E+01$	$1.92E+01$
		Worst	**$2.39E+01$**	$1.59E+02$	$2.92E+01$	$9.35E+01$
		Mean	**$2.05E+01$**	$3.35E+01$	$2.44E+01$	$3.05E+01$
		Median	**$2.08E+01$**	$2.88E+01$	$2.48E+01$	$2.44E+01$
		StDev	**$2.21E+00$**	$3.42E+01$	$2.50E+00$	$1.70E+01$
F_5	600000	Best	$1.39E+01$	$4.08E+01$	**$1.09E+01$**	$2.51E+02$
		Worst	**$3.28E+01$**	$1.78E+02$	$3.38E+01$	$3.72E+02$
		Mean	$2.30E+01$	$7.87E+01$	**$2.24E+01$**	$3.25E+02$
		Median	**$2.09E+01$**	$6.87E+01$	$2.19E+01$	$3.26E+02$
		StDev	**$5.33E+00$**	$3.53E+01$	$6.87E+00$	$2.99E+01$
F_6	300000	Best	$9.31E-01$	$2.15E+00$	**$3.48E-06$**	$1.39E+01$
		Worst	$2.41E+00$	$3.92E+00$	**$2.01E+00$**	$1.61E+01$
		Mean	$1.72E+00$	$3.06E+00$	**$1.10E+00$**	$1.52E+01$
		Median	$1.78E+00$	$3.01E+00$	**$1.50E+00$**	$1.54E+01$
		StDev	**$4.01E-01$**	$6.64E-01$	$7.73E-01$	$5.90E-01$

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported in part by the US National Science Foundation's BEACON Center for the Study of Evolution in Action, funded under Cooperative Agreement no. DBI-0939454.

References

[1] I. Fister Jr., D. Fister, and X. S. Yang, "A hybrid bat algorithm," *Elektrotehniški Vestnik*, vol. 80, no. 1-2, pp. 1–7, 2013.

[2] W. L. Goffe, G. D. Ferrier, and J. Rogers, "Global optimization of statistical functions with simulated annealing," *Journal of Econometrics*, vol. 60, no. 1-2, pp. 65–99, 1994.

[3] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

[4] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, vol. 1, pp. 39–43, Nagoya, Japan, October 1995.

[5] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[6] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 687–697, 2008.

[7] X. S. Yang, "Firefly algorithm," in *Engineering Optimization*, pp. 221–230, John Wiley & Sons, New York, NY, USA, 2010.

[8] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimization," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[9] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, vol. 284 of *Studies in Computational Intelligence*, pp. 65–74, Springer, Berlin, Germany, 2010.

[10] B. Y. Qu, P. N. Suganthan, and J. J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization,"

IEEE Transactions on Evolutionary Computation, vol. 16, no. 5, pp. 601–614, 2012.

[11] G. Wang and L. Guo, "A novel hybrid bat algorithm with harmony search for global numerical optimization," *Journal of Applied Mathematics*, vol. 2013, Article ID 696491, 21 pages, 2013.

[12] H. Xu, C. Caramanis, and S. Mannor, "Sparse algorithms are not stable: a no-free-lunch theorem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 187–193, 2012.

[13] S. Sanner and C. Boutilier, "Approximate linear programming for first-order MDPs," in *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI '05)*, pp. 509–517, Edinburgh, UK, July 2005.

[14] S. Yazdani, H. Nezamabadi-Pour, and S. Kamyab, "A gravitational search algorithm for multimodal optimization," *Swarm and Evolutionary Computation*, vol. 14, pp. 1–14, 2014.

[15] G. Obregon-Henao, B. Babadi, C. Lamus et al., "A fast iterative greedy algorithm for MEG source localization," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '12)*, pp. 6748–6751, San Diego, Calif, USA, September 2012.

[16] R. S. Parpinelli and H. S. Lopes, "New inspirations in swarm intelligence: a survey," *International Journal of Bio-Inspired Computation*, vol. 3, no. 1, pp. 1–16, 2011.

[17] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, 2010.

[18] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the Congress on Evolutionary Computation*, pp. 81–86, IEEE, Seoul, South Korea, May 2001.

[19] X.-S. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[20] P.-W. Tsai, J.-S. Pan, B.-Y. Liao, M.-J. Tsai, and V. Istanda, "Bat algorithm inspired algorithm for solving numerical optimization problems," *Applied Mechanics and Materials*, vol. 148-149, pp. 134–137, 2012.

[21] X.-S. Yang and X. He, "Bat algorithm: literature review and applications," *International Journal of Bio-Inspired Computation*, vol. 5, no. 3, pp. 141–149, 2013.

[22] I. Fister, S. Fong, J. Brest, and I. Fister, "A novel hybrid self-adaptive bat algorithm," *The Scientific World Journal*, vol. 2014, Article ID 709738, 12 pages, 2014.

[23] H. Liu, F. Gu, and X. Li, "A fast evolutionary algorithm with search preference," *International Journal of Computational Science and Engineering*, vol. 3/4, no. 3-4, pp. 197–212, 2012.

[24] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Advances in Swarm Intelligence*, pp. 355–364, Springer, Berlin, Germany, 2010.

[25] X. S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*, pp. 240–249, Springer, Berlin, Germany, 2012.

[26] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245–276, 1996.