ELSEVIER

Advances in Engineering Software 37 (2006) 106–111

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

A new optimization method: Big Bang–Big Crunch

Osman K. Erol*, Ibrahim Eksin

Electrical & Electronics Engineering Faculty, Istanbul Technical University, Maslak, 34390 Istanbul, Turkey

Received 10 May 2004; received in revised form 21 February 2005; accepted 21 April 2005
Available online 18 July 2005

Abstract

Nature is the principal source for proposing new optimization methods such as genetic algorithms (GA) and simulated annealing (SA) methods. All traditional evolutionary algorithms are heuristic population-based search procedures that incorporate random variation and selection. The main contribution of this study is that it proposes a novel optimization method that relies on one of the theories of the evolution of the universe; namely, the Big Bang and Big Crunch Theory. In the Big Bang phase, energy dissipation produces disorder and randomness is the main feature of this phase; whereas, in the Big Crunch phase, randomly distributed particles are drawn into an order. Inspired by this theory, an optimization algorithm is constructed, which will be called the Big Bang–Big Crunch (BB–BC) method that generates random points in the Big Bang phase and shrinks those points to a single representative point via a center of mass or minimal cost approach in the Big Crunch phase. It is shown that the performance of the new (BB–BC) method demonstrates superiority over an improved and enhanced genetic search algorithm also developed by the authors of this study, and outperforms the classical genetic algorithm (GA) for many benchmark test functions.

Keywords: Big Bang–Big Crunch evolution theory; Evolutionary algorithms; Genetic algorithm

1. Introduction

The optimization problem can be summarized as finding the set of parameters $\{x_n\}$ that minimizes an objective function $f(x_0,\ldots,x_n)$ that is also referred to as the fitness function in evolutionary algorithms. Global optimization requires the arrival to the best possible decision in any given set of circumstances. Genetic algorithms (GA) [1–4] are a class of stochastic and global optimization which model the basic principles and mechanisms of Mendelian genetics and evolutionary theory along with population genetics to tackle hard search and optimization problems. The main difference between the GA and the other classical search algorithms is that the GA operates on a 'population' of strings (chromosomes in genetic language) instead of one input parameter. This gives the GA the ability to globally search optimum parameter. However, the GA suffers from premature convergence, convergence speed and execution time problems. To overcome these difficulties, many publications on the GA and its associated operators such as roulette-wheel selection [1], byte-wise crossover [5], combined mutation operators [6], or Cauchy mutations [7] have been proposed.

Classical GA when coupled with a highly elitist selection property have the potential to gather all of the population to the fittest member; that is, the GA may fail to converge to the global optimum point depending upon the selective capacity of the fitness function (large variation of the fitness function for small variation of its input variables). In other words, a fitness function is called elitist if it exaggerates the small differences among the fitness values of the chromosomes. For instance, a hyperbolic function is more elitist than a linear function. The global convergence property of the algorithm is assured only through the mutation operator, which in turn creates new chromosomes at the expense of extra computational effort. Moreover, if the response surface is quite smooth then the GA requires more response (fitness) function evaluations compared to classical hill-climbing techniques. Some 'more intelligent' rules and/or hybrid techniques such as evolutionary-gradient search (EGS) have been added to the GA to overcome this slow convergence phenomena of the GA near the optimum solution [8–11].

* Corresponding author.

0965-9978/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2005.04.005

A priori knowledge may be added in the initialization stage as well. The straightforward approach is to take samples uniformly at random from a state space of possible solutions, thereby providing an unbiased initial population. This is often useful in benchmarking evolutionary algorithms, but when solving real problems, some information that would help to seed the initial population with useful hints is known. Biasing the initial population towards better solutions can often save a great deal of time that would otherwise be spent 'reinventing the wheel' [12]. However, all of these precautions cause the algorithms to be inflated by extra number of calculations.

In this study, the difference between classical genetic algorithms and combat genetic algorithm are first briefly explained and then a review of an improved and enhanced genetic search algorithm developed and named as the 'combat genetic algorithm' (C-GA) by the authors of this study [13] is reconsidered in Section 2. The Big Bang–Big Crunch (BB–BC) method, which is the main contribution of this study, is presented in Section 3. Next, the results of the Big Bang–Big Crunch method are compared with the results of the combat genetic algorithm (C-GA) in Section 4. Finally, the overall comparison results and the advantages of the proposed method are discussed in Section 5.

2. Genetic operators and combat genetic algorithm

The C-GA differs from classical GA in the respect that the diversity is kept high by reducing the aggressive convergence tendency of the reproduction operator. Although the selection or reproduction operators in GA are the principal elements that shift the chromosomes towards a local/global optimum point, due to the decreasing diversity these can be viewed as a source of premature convergence to a local optimum point. In both GA and C-GA, the increase of meaningful information is assured by creating new members whether by using crossover or applying mutation. Crossover and mutation can be viewed as an energy intrusion into the algorithm. However, if all the chromosomes become similar because of the reproduction operator, the crossover will not be able to produce new chromosomes. Therefore, the modifications done on the reproduction stage of classical GA is very important and it will be quite effective in performance enhancement. The C-GA combines two basic operators of the GA's; namely, reproduction and crossover and reduces the amount of calculations greatly compared to classical GA's.

The C-GA algorithm can be summarized as follows:

Step 1 Form an initial generation of m chromosomes in a random manner.

Step 2 Choose two chromosomes randomly from the initially generated pool.

Step 3 Calculate the fitness values for the two randomly selected chromosomes and then form the relative difference by using

$$\Delta_r = \frac{|f_1 - f_2|}{f_1 + f_2} \tag{1}$$

where f_1 and f_2 are the fitness values of the chromosome$_1$ and chromosome$_2$, respectively.

Step 4 Compare the fitness values of the two selected chromosomes. If there is a relatively big difference between the fitness values, then the partial overwrite operator is applied. This means that the less-fit chromosome loses some information. If the difference is relatively small then the classical uniform crossover operation is applied. In mathematical terms:

If $f_1 < f_2$ and $\lambda < \Delta_r$, where λ represents a random number selected between [0, 1] then perform crossover on the second chromosome only and leave the first chromosome unchanged since it is better than the second one if minimization is considered. However, if $f_1 < f_2$ and $\lambda > \Delta_r$, perform normal crossover.

If $f_1 > f_2$ and $\lambda < \Delta_r$, then apply crossover only on the first individual and leave the second chromosome unchanged. However, if $f_1 > f_2$ and $\lambda < \Delta_r$, then apply normal crossover.

Step 5 Apply mutation operator with a probability of $1/m$.

Step 6 Return to Step 2 until a stopping criterion has been accomplished.

Steps 3-4 are similar to selection and crossover, but their appearance order is regulated by a dominance factor given in Eq. (1).

3. Big Bang–Big Crunch (BB–BC) method

Randomness can be seen as equivalent to the energy dissipation in nature while convergence to a local or global optimum point can be viewed as gravitational attraction. Since energy dissipation creates disorder from ordered particles, we will use randomness as a transformation from a converged solution (order) to the birth of totally new solution candidates (disorder or chaos).

The proposed method is similar to the GA in respect to creating an initial population randomly. The creation of the initial population randomly is called the Big Bang phase. In this phase, the candidate solutions are spread all over the search space in an uniform manner. Fig. 1 is drawn to give an idea how the candidate solutions are spread in the optimization problem of Rosenbrock function [11] with two 2-bytes long integer variables symbolizing the real values between [0, 10].

Since the normal random number generator can produce numbers greater than unity, it is therefore necessary to limit their values in order to keep them in the search space. The effect of this limitation can be seen in Fig. 1 as an

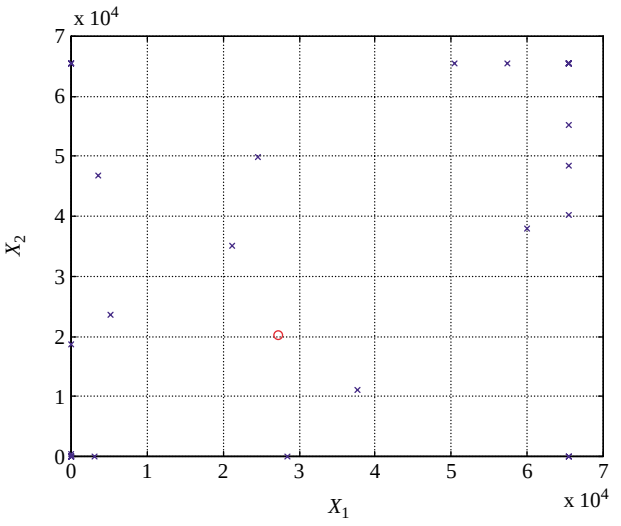

Fig. 1. Initial spread of candidate solutions for two-dimensional case where the parameters are denoted by 'x_1 and x_2' and their respective center of mass is marked with 'o' for the Rosenbrock function.

accumulation of candidates at the boundaries. The population size is kept fixed as 30 in the example and the benchmark tests. However, this size can be reduced or increased according to the convergence or the number of iterations.

The Big Bang phase is followed by the Big Crunch phase. The Big Crunch is a convergence operator that has many inputs but only one output, which can be named as the center of 'mass', since the only output has been derived by calculating the center of mass. Here, the term mass refers to the inverse of the fitness function value. The point representing the center of mass that is denoted by x^c is calculated according to the formula:

$$\vec{x}^{c} = \frac{\sum_{i=1}^{N} \frac{1}{f^i} \vec{x}^i}{\sum_{i=1}^{N} \frac{1}{f^i}} \tag{2}$$

where x^i is a point within an n-dimensional search space generated, f^i is a fitness function value of this point, N is the population size in Big Bang phase. The convergence operator in the Big Crunch phase is different from 'exaggerated' selection since the output term may contain additional information (new candidate or member having different parameters than others) than the participating ones, hence differing from the population members. This one step convergence is superior compared to selecting two members and finding their center of gravity [14,15]. This method takes the population members as a whole in the Big-Crunch phase that acts as a squeezing or contraction operator; and it, therefore, eliminates the necessity for two-by-two combination calculations.

After the Big Crunch phase, the algorithm must create new members to be used as the Big Bang of the next iteration step. This can be done in various ways, the simplest one being jumping to the first step and creating an initial population. The algorithm will have no difference than random search method by so doing since latter iterations will not use the knowledge gained from the previous ones; hence, the convergence of such an algorithm will most probably be very low. An optimization algorithm must converge to an optimal point; but, at the same time, in order to be classified as a global algorithm, it must contain certain different points within its search population with a decreasing probability. To be more precise, we mean that, large amount of solutions generated by the algorithm must be around the 'so-called' optimal point but the remaining few points in the population bed must be spread across the search space after certain number of steps. This ratio of solution points around the optimum value to points away from optimum value must decrease as the number of iterations increases; but, in no case, it could be equal to zero, which means the end of the search. This convergence or the use of the previous knowledge (center of mass) can be accomplished by spreading new off-springs around this center of mass using a normal distribution operation in every direction where the standard deviation of this normal distribution function decreases as the number of iterations of the algorithm increases. The new members are shown in Fig. 2. It is obvious that only two variable cases can be illustrated in figures while theoretically there is no limit on the dimension of the search space, which are 10 in our test experiments. This convergence can be formulated as below, where the space boundary is the sum of the Euclidian distances of all members:

$$\text{Space boundary in the } k\text{th iteration/space boundary in the } (k+1)\text{th iteration} > 1 \tag{3}$$

After the second explosion, the center of mass is recalculated. These successive *explosion* and *contraction*

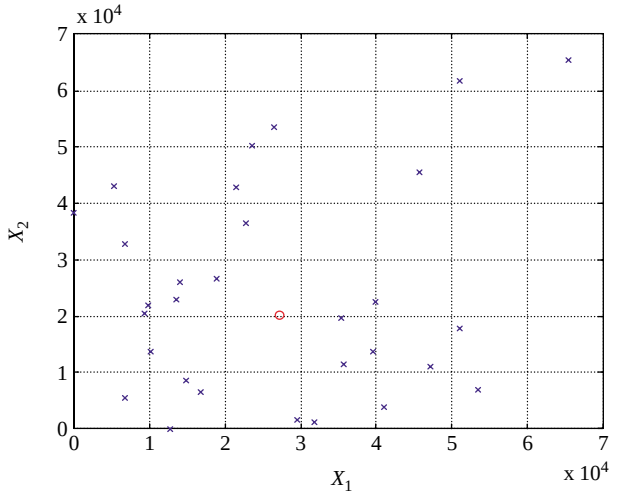

Fig. 2. Spread of candidate solutions for two-dimensional case where the parameters are denoted by 'x_1 and x_2' and their respective center of mass marked with 'o' for the Rosenbrock function after the 4th iteration.

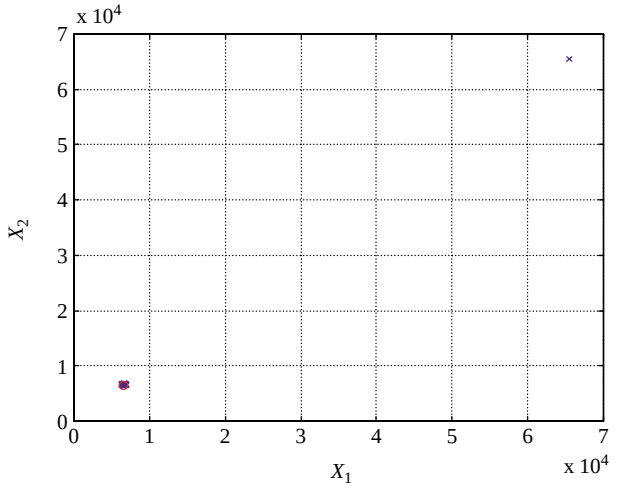

Fig. 3. Accumulation of candidate solutions on the center of mass after the 500th iteration.

steps are carried repeatedly until a stopping criterion has been met. The two parameters to be supplied to normal random point generator are the center of mass of the previous step and the standard deviation. The deviation term can be fixed, but decreasing its value along with the elapsed iterations produces better results as given in Fig. 3. The accumulation around the center of mass can be seen, but there still exist a few points out of the ensemble.

The accumulation point of candidate solutions as is seen in Fig. 3, [6553, 6553] represents the decimal value [1, 1]. Moreover, this point is the optimal solution of the Rosenbrock function. In Fig. 3, nearly all the 30 members lie on the optimal point, but there still exists one more point far beyond the optimal solution. The deviation term will reach zero as iterations go to infinity, thus we can conclude there will always be off-springs which will be located far from the center of mass with decreasing probability but never equal to zero bearing the potential to affect the so-found center of mass towards itself if it has higher fitness value than the remaining members. This is the key property that assures the global convergence of the algorithm.

Summarizing the steps in BB–BC yields to:

Step 1 Form an initial generation of N candidates in a random manner. Respect the limits of the search space.

Step 2 Calculate the fitness function values of all the candidate solutions.

Step 3 Find the center of mass according to Eq. (2). Best fit individual can be chosen as the center of mass instead of using Eq. (2).

Step 4 Calculate new candidates around the center of mass by adding or subtracting a normal random number whose value decreases as the iterations elapse. This can be formalized as

$$x^{\text{new}} = x^{\text{c}} + lr/k \tag{4}$$

where x^{c} stands for center of mass, l is the upper limit of the parameter, r is a normal random number and k is the iteration step. Then new point x^{new} is upper and lower bounded.

Step 5 Return to Step 2 until stopping criteria has been met.

4. Benchmark test results

The new algorithm BB–BC has been tested and compared with the C-GA on the benchmark problems taken from [11]. This list comprises some widely used test functions such as sphere, Rosenbrock, step, ellipsoid, Rastrigin and Ackley functions given in Table 1.

In these experiments, the normalized performance measure which is given by

$$\log(f^*) = \log\left(\frac{f(\vec{x}^0)}{f(\vec{x}^*)}\right) \tag{5}$$

is used. Here, $\vec{x}^0$ is chosen as [10, 10, 10,..., 10] to be a reference point. Each of the ameliorations is calculated for 30 independent trials and the average of the best fitness values is taken. The population size m, the dimension n, and the resolution or the number of bits for each parameter is chosen to be equal to 30, 10, and 16, respectively. The maximum value that a parameter (x_is in these cases) can take is limited to 10.

In BB–BC method, the number of iterations is limited to 500 while the number of independent trials as well as other parameters remains unchanged. The results are shown in Fig. 4. In order to make a fair comparison between the two algorithms, 500 iterations are chosen as stopping criteria in the simulations. One selection, crossover and mutation operation in the combat approach is considered to be a single iteration.

It can be seen that the BB–BC method finds the optimum solution in finite steps, which makes the amelioration infinite in cases (a), (c), (d) and (e) in Fig. 4. It is also seen from the same figure that the BB–BC method cannot find the exact solution for the Rosenbrock and Ackley functions within 500 iterations. The same fact is also valid for the C-GA. Even though, the BB–BC method seems to be slow compared to the C-GA in approximately the first 10

Table 1
The function test bed

Function	Name
$f_{\text{sphere}}(\vec{x}) = \sum_{i=1}^{n} x_i^2 = \|\|\vec{x}\|\|$	Sphere function
$f_{\text{rosen}}(\vec{x}) = \sum_{i=1}^{n-1}[100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2]$	Rosenbrock function
$f_{\text{step}}(\vec{x}) = \sum_{i=1}^{n}[\|x_i\| + 0.5]^2$	Step function
$f_{\text{elli}}(\vec{x}) = \sum_{i=1}^{n} i x_i^2$	Ellipsoid function
$f_{\text{rast}}(\vec{x}) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	Rastrigin function
$f_{\text{ackley}}(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	Ackley function

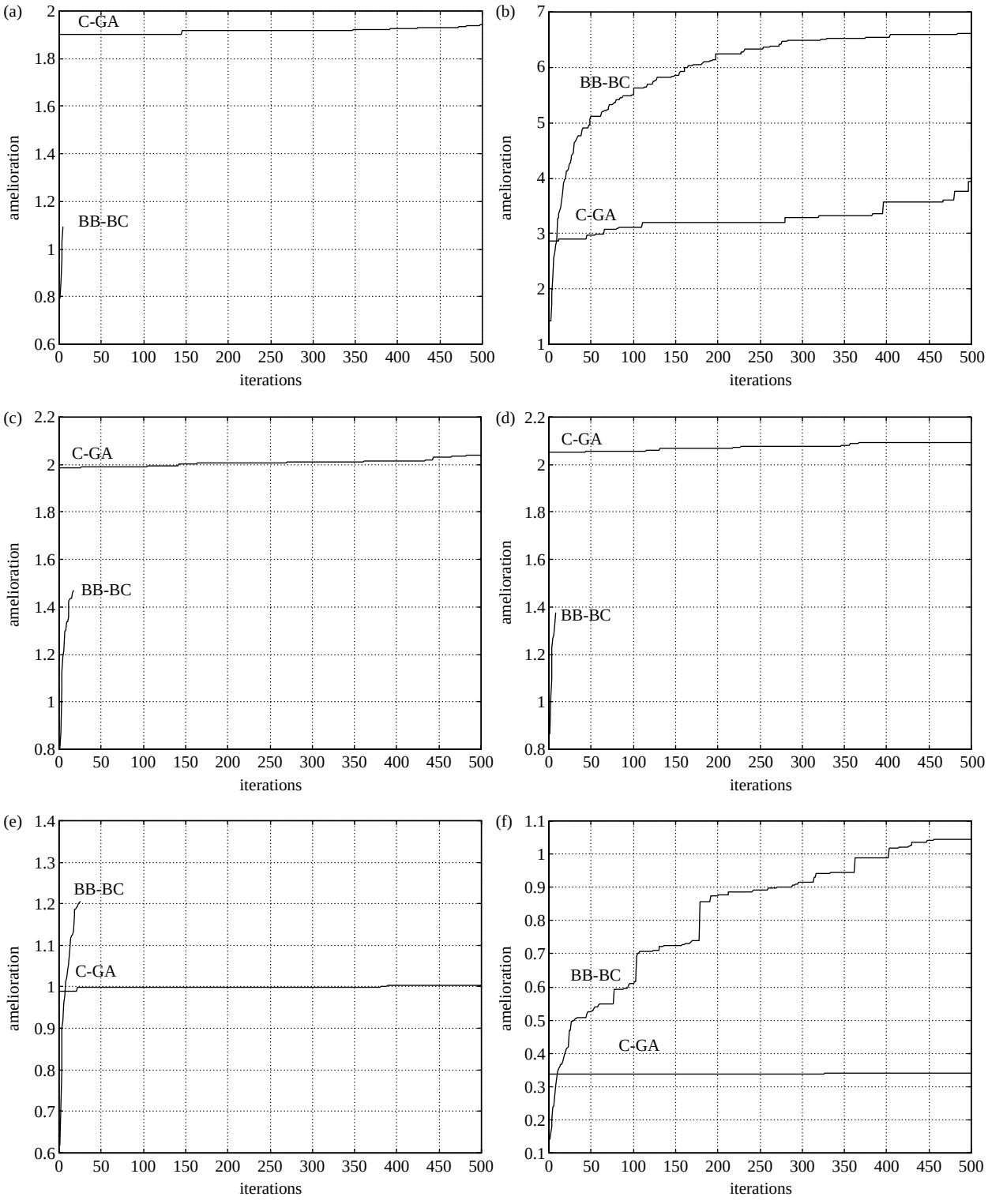

Fig. 4. Comparison of the results of the BB–BC method with C-GA on (a) spherical, (b) Rosenbrock, (c) step, (d) ellipsoid, (e) Rastrigin, and (f) Ackley test functions.

iterations, the new BB–BC method still outperforms the C-GA in the cases of the Rosenbrock and Ackley test functions.

The new method consumes 2.27 times more CPU time than the C-GA does for the same amount of iterations and equal parameters. This drawback of the new method should be compensated by increasing the number of iterations in the C-GA so as to make a number of function evaluations the same to have a fair comparison between the two approaches. In that case, the results that are illustrated in

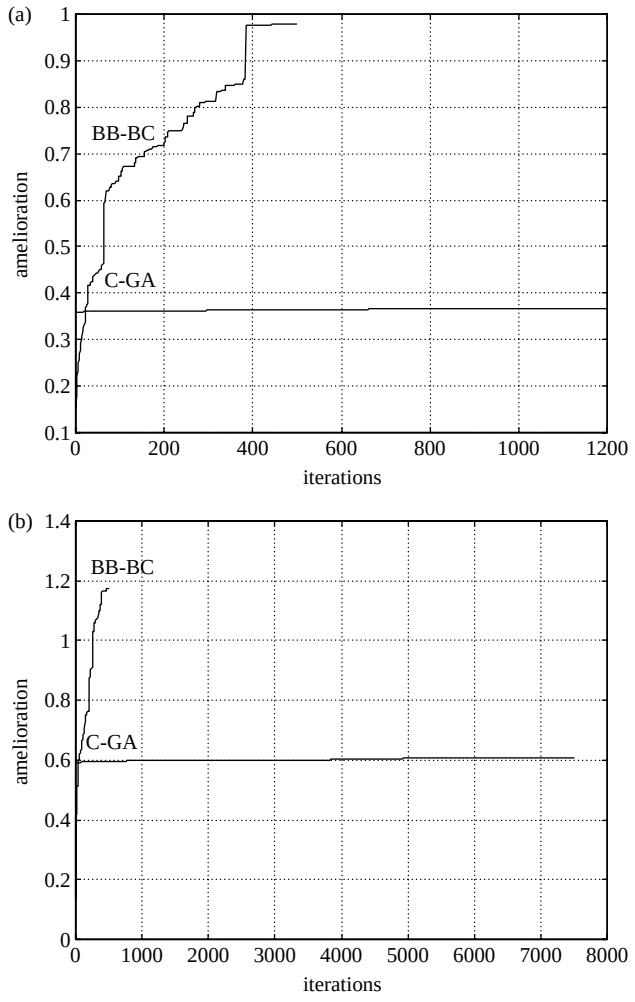

Fig. 5. Time and function evaluation compensated comparisons on Ackley function. (a) Time compensated. (b) Function evaluation compensated.

Fig. 5(a) have been obtained. If one considers the number of function evaluations at each iteration step, the new method takes into account 30 members while the C-GA carries operations on two chromosomes only. Therefore, in order to make a fair comparison between the two approaches one must again increase the number of iterations by a factor of 15 in favor of the C-GA. In that case, the result shown in Fig. 5(b) has been obtained. In both considerations, it has been observed that the BB–BC method still ameliorates much faster than the C-GA.

5. Conclusions

When searching for the global optimum of complex problems, particularly in problems with many local optima, the main conflict is between 'accuracy', 'reliability' and 'computation time'. In the case that traditional optimization methods fail to provide efficiently reliable results, evolutionary algorithms may constitute an interesting alternative. Nevertheless, classical GA or its variant combat approach may still suffer from excessively slow convergence; that is, they are generally sluggish in reaching the global optimum accurately and reliably in a short period of time. However, they have the ability to rough out a problem 'reliably' by finding the most promising spaces in the entire search space.

In this study, it has been shown that the main drawback of the classical GA might be overcome by the new BB–BC method; that is, the speed of convergence has been ameliorated beyond comparison in the benchmark test problems given in Table 1 with respect to C-GA. Moreover, another conclusion that can be made after the analysis of the results of the various methods on the test functions is that the amelioration given in (5) depends also on the nature of the function to be minimized and as the function becomes more selective, the speed of convergence decreases for both approaches. However, the BB–BC method finds the exact global optimum point for the sphere, step, Rastrigin functions within the maximum number of allowed iterations; while it still outperforms the C-GA method for the Rosenbrock and Ackley functions.

References

[1] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley; 1989.
[2] Holland JH. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. 2nd ed. Cambridge: MIT Press; 1992.
[3] Ed LD. Handbook of genetic algorithms. New York: Van Nostrand Reinhold; 1991.
[4] Bäck T, Hammel U, Schwefel HP. Evolutionary computation: comments on the history and current state. IEEE Trans Evol Comput 1997;1(1):3–17.
[5] Davidor Y. Genetic algorithms and robotics. World Sci Ser Rob Autom Syst 1991;1:1–41.
[6] Chellapilla K. Combining mutation operators in evolutionary programming. IEEE Trans Evol Comput 1998;2(3):91–6.
[7] Rudolph G. Local convergence rates of simple evolutionary algorithms with Cauchy mutations. IEEE Trans Evol Comput 1997; 1(4):249–58.
[8] Glover F. Tabu-search. Part I. ORSA J Comput 1989;1(3): 190–206.
[9] Karaboğa D, Kaplan A. Optimizing multivariable functions using tabu search algorithm. The 10th International Symposium on computational and Information sciences ISCIS X, Turkey, October, vol. 2. p. 793–9.
[10] Kim JH, Myung H. Evolutionary programming techniques for constrained optimization problems. IEEE Trans Evol Comput 1997; 1(2):129–40.
[11] Salomon R. Evolutionary algorithms and gradient search: similarities and differences. IEEE Transa Evol Comput 1997;2(2):45–55.
[12] Chou CH, Chen JN. Genetic algorithm's initialization schemes and genes extraction. Proceedings of ninth IEEE international conference on fuzzy systems, vol. 2, p. 965–8.
[13] Eksin I, Erol OK. Evolutionary algorithm with modifications in the reproduction phase. IEE Proc Softw 2001;148(2):75–80.
[14] Stuckman B. A global search method for optimizing nonlinear systems. IEEE Trans Syst, Man, Cybernet 1988;18(6):965–77.
[15] Tigli O, Eksin I, Ertem S, Boz O. A global optimization in controlling a plant. Electr Power Syst Res 1994;30(3):257–62.