Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 1359 – 1375

doi:10.3906/elk-1106-46

Big bang-big crunch optimization algorithm with local directional moves

Hakkı Murat GENÇ,* İbrahim EKSİN, Osman Kaan EROL
Department of Control Engineering, Faculty of Electrical and Electronics Engineering,
İstanbul Technical University, 34469, İstanbul, Turkey

Received: 30.06.2011 • **Accepted:** 04.04.2012 • **Published Online:** 12.08.2013 • **Printed:** 06.09.2013

Abstract: The big bang–big crunch (BB–BC) algorithm has been proposed as 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. It has been shown that this algorithm is capable of quick convergence, even in long, narrow parabolic-shaped flat valleys or in the existence of several local extremes. In this work, the BB–BC optimization algorithm is hybridized with local search moves in between the "banging" and "crunching" phases of the algorithm. These 2 original phases of the BB–BC algorithm are preserved, but the representative point ("best" or "fittest" point) attained after the crunching phase of the iteration is modified with some local directional moves, using the previous representative points so far attained, with the hope that a better representative point would be obtained. The effect of imposing local directional moves is inspected by comparing the results of the original and enhanced BB–BC algorithm on various benchmark test functions. Moreover, the crunching phase of the algorithm is improved with the addition of a simplex-based local search routine. The results of the new hybridized algorithm are compared with the state-of-the-art variants of widely used evolutionary computation methods, namely genetic algorithms, covariance matrix adaptation evolutionary strategies, and particle swarm optimization. The results over the benchmark test functions have proven that the BB–BC algorithm, enhanced with local directional moves, has provided more accuracy with the same computation time or for the same number of function evaluations.

Key words: Big bang–big crunch optimization algorithm, evolutionary computation, hybrid local search

1. Introduction

Researchers throughout the world work on global optimization problems in numerous research and development fields. In global optimization problems, the objective function is the key point that identifies the problem. Because of the different characteristics of the objective functions, each global optimization algorithm may not work with the same performance in all of the problem domains. This fact forces us to propose new methods for new problems. As well as constructing brand new optimization algorithms, hybridizing existing methods to solve a variety of objective functions with a unique method is still a great concern for researchers.

Global optimization algorithms are computing techniques that are mostly inspired from the natural processes. The big bang and big crunch (BB–BC) algorithm [1] is also a global optimization method relying on heuristics from nature, particularly, the theory of the big bang and the big crunch. The algorithm generates new candidate solutions randomly in the big bang phase and those solution candidates are used to obtain a single representative point through a contraction approach in the big crunch phase. The BB–BC algorithm has been applied in many areas, including fuzzy model inversion [2,3], the design of space trusses [4], the size

*Correspondence: hmuratgenc@gmail.com

reduction of space trusses [5], the airport gate assignment problem [6], nonlinear controller design [7], and genetic programming classifier design [8].

Memetic algorithms (MAs) represent one of the recent growing areas of research in evolutionary computation. The first use of the term "memetic algorithms" in the computing literature appeared in 1989 [9]. The rationale behind MAs is to provide an effective and efficient global optimization method by compensating for the deficiency of evolutionary algorithms (EA) in local exploitation and the inadequacy of local search (LS) in global exploration [10]. The term MA is now widely used for any population-based approach with separate local improvement procedures.

Real coded memetic algorithms can be classified into 2 main classes, depending on the type of LS employed [11]:

1) Local improvement process (LIP) oriented LS (LLS): This category refines the solutions of each generation by applying efficient LIPs, like gradient descent. LIPs can be applied to every member of the population or with some specific probability and with various replacement strategies.

2) Crossover-based LS: This group employs crossover operators for local refinement. A crossover operator is a recombination operator that produces offspring around the parents. For this reason, it may be considered as a move operator in an LS strategy [11].

The adaptation of the parameters has become a very promising research field in MAs. Ong and Keane [12] proposed meta-Lamarckian learning in MAs that adaptively chooses among multiple memes during a MA search. They proposed 2 adaptive strategies in their work and their empirical studies showed their superiority over other traditional MAs. A taxonomy and comparative study on the adaptive choice of the memes in MAs was presented in [13]. In order to balance between the local and genetic search, Bambha et al. proposed simulated heating that systematically integrates parameterized LS (both statically and dynamically) into EAs [14]. Ahn et al. also applied an adaptive local search routine to multiobjective evolutionary optimization problems [15]. The common aspect for all of the memetic methods proposed so far is that they needed mechanisms that have to:

i. decide the step length (and adaptation of step length) of the local search, and

ii. draw a balance between exploration and exploitation; that is, local search and global search.

A comprehensive review on hybrid genetic algorithms can be found in [16].

In this study, a new memetic algorithm is introduced in which a local search is imposed between the phases of the BB–BC optimization method. The local search algorithm generates a direction vector using the current fittest point and the previous fittest points of the generations and checks for an improvement in this direction. If an improvement is achieved, the new center is forced to switch to that point. That is to say, the center point of the explosion of the next big bang phase is changed. Note that, by using the distance between these consecutive representative points, the step size of the local search is set and adjusted accordingly. The exploitation or intensification capability of the algorithm is enhanced with local search; thus, the proposed hybridization operation produces much more accurate results than the original BB–BC algorithm. In fact, it also provides promising results when compared to the state-of-the-art optimization methods. Moreover, the newly proposed algorithm is shown to be much more effective in terms of complexity.

The rest of the paper is divided into 4 sections. The BB–BC algorithm and the Nelder–Mead (NM) crunching phase are summarized in Section 2. The proposed hybridization algorithm is detailed in Section 3.

In Section 4, the simulation results on various test functions are presented to illustrate the effectiveness of the new hybrid algorithm. Possible further developments and conclusions are finally elaborated on and discussed in Section 5.

2. BB–BC optimization algorithm

The BB–BC algorithm is a population-based evolutionary computation method. The algorithm is shown to be fast convergent, both in unimodal and multimodal topologies. In this section, a brief summary of the BB–BC algorithm is given and the new crunching operator is introduced. Further details of the algorithm can be found in [1].

Aside from the random initial population, in each iteration the population is created in the big bang phase. In this phase, the candidate solutions are spread over the search space using normal distribution. The standard deviation of the distribution is adjusted with respect to the iteration count and the mean value of the distribution is taken to be the last representative point found. The impermissible candidates are handled at this phase. Once the new population is created, the candidate individuals are assigned a fitness value to be used in the following big crunch phase. Big crunch is a multiple input–single output operator. The inputs are the whole set of individuals forming the current population and the output is the representative point of the iteration or "center of mass". The representative point can be calculated in 3 ways:

i. By weighting the individuals with corresponding fitness evaluations, as in Eq. (1).

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

In Eq. (1), $\vec{x}^i$ is the position vector for the ith individual and f^i stands for the fitness value of the ith individual.

ii. The fittest individual can be selected as the center of mass.

iii. Crunching can be performed as the result of NM optimization method.

NM is a simple yet effective local search optimization method. The idea is to use the selected population members to generate a $(D+1)$ dimensional simplex and end up with a local minimum as the center of mass, where D is the search space dimension. The vertex with the worst function value of the $(D+1)$ – simplex is rejected and replaced with a new vertex. Using this method as the big crunch operator imposes the following constraint on population size N:

$$N \geq D+1. \tag{2}$$

The stopping criterion for the NM crunching phase needs to be decided as the function evaluation (FE) budget of the algorithm (nm_b) or the termination tolerance (nm_t), depending on attribute or objective space distances.

NM crunching is a new approach in order to improve the exploitation capability near a local minimum. If NM crunching is used at the initial iterations of the search, then excessive FEs have to be performed, since there is not enough knowledge on the function topology or coverage. Next, it is better not use

NM crunching at the early iterations of the search algorithm; instead, this crunching method should be switched when the search has evolved and ripened. The effect of NM crunching after the initial iterations is illustrated in the convergence graph in Figure 1 for a multimodal function.

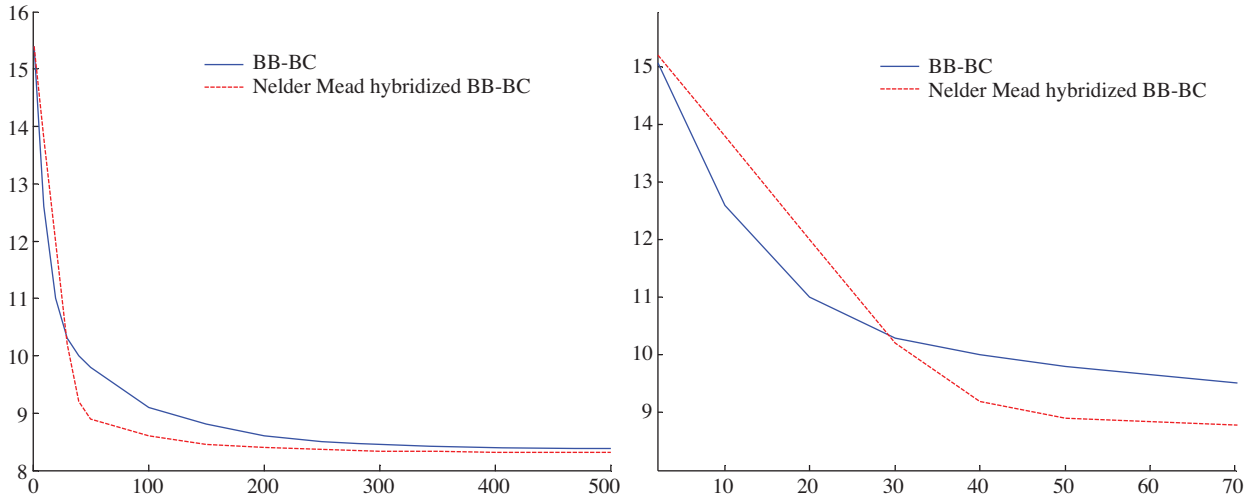

Figure 1. Convergence graphs for BB-BC algorithms using the best point as the center of mass (solid line) and the NM method (dashed line).

After the crunching phase, new candidates are calculated 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:

$$\vec{x}^{new} = \vec{x}^{c} + \frac{R.r}{1 + k/sm}. \tag{3}$$

In Eq. (3), x^c is the center of mass, R is the search range, r is a normal random number, k is the iteration step, and sm is the parameter adjusting the explosion strength. The algorithm continues until a predefined stopping criterion has been met [1].

3. BB–BC with local directional moves (BBBC–LS)

The proposed hybrid method has basically 2 main parts, namely the global search and the local search parts following each other. The global search part of the algorithm is the BB–BC algorithm that is reviewed in the previous section and the algorithm has been reserved and applied with no modification within itself or its parameters. Within the local search part of the algorithm, a search line (direction) is defined using the current and the previous fittest member of the generation of the BB–BC global search optimization algorithm (center of mass) and the local search is accomplished in the direction of the search within the neighboring points.

The steps of the algorithm can be summarized as follows:

1. Form an initial generation of N individuals in a random manner.
2. Perform the crunching phase of the BB–BC algorithm. The center of mass is selected as the fittest individual. This point becomes the first best point found in the iteration. Store this point.

3. Perform once more the consecutive banging and then the crunching phases of the BB–BC algorithm. The crunching phase is switched to NM crunching after the T_{fe} portion of the FEs is completed.

4. If the "best" point obtained in step 3 is better than the last stored point, then this means an improvement; then store that point. Next, generate a direction vector using 1 or 2 previous "best" candidate solution points so far attained, and make n_h exploratory moves in that direction and assign a new virtual center of mass on that direction if a better point has been obtained than the previous fittest points. If the best (fittest) point remains the same after the local search phase, then go straight to step 5.

5. Check the stopping criterion. If it is met, stop; otherwise, go back to step 3.

The definition, abbreviation, and value intervals for the algorithm specific parameters are listed in Table 1.

The proposed idea in this study is to speed up the search by checking some local points after the crunching phase of the main global search algorithm, so as to maximize the improvement probability. The neighboring points' check procedure should be carried out in the guided and limited direction(s). Otherwise, checking random neighbors or a complete set of neighbors can cause an unacceptable processing time or even search stagnation. In this study, the proposed local search moves of the hybridization procedure are based on defining a possible improving direction to check the neighboring points. Between iterations, the movement of the best point forms a basis for the linear search direction definition. Figure 2 gives the flowchart for the algorithm in a generic manner. Search directions are generated by utilizing autoregression on the locations of the representative points of consecutive crunch phases. The local search operation can be performed for a few predetermined numbers of steps on these directions, so abstaining from sticking into a local optimum point. Any local search method can be utilized in these generated directions; here, the expansion and contraction moves of the basic simplex search method and dichotomous search algorithms are exploited for the local search phase of the hybridized optimization method.

Table 1. Definitions of the algorithm parameters.

Symbol	Definition	Value interval	Data type
N	Population size	$N \geq D+1$, if NM crunching is used	integer
nm_b	Allowed FE budget for the NM crunching phase (if used)	$nm_b <$ total FE budget	integer
Nm_t	NM crunching tolerance error (if used), (algorithm ends either nm_b or nm_t fulfilled)	$0 < nm_t < \infty$	double
n_h	Number of expansion/contraction steps performed between each iteration	$0 < n_h <$ total FE budget	integer
T_{fe}	Normalized crunching phase switching parameter: After T_{fe} proportion of total FE carried switch to NM crunching	$0 \leq T_{fe} \leq 1$	double
sm	Explosion strength adjusting parameter, determines mean step size of banging phase	$1 \leq sm \leq \infty$	double

Three methodologies are developed for the local improvement of the BB–BC algorithm and the details are given below.

3.1. Direction vector formation with single step regression

The direction vector is simply the difference between the vector of the fittest point of the current iteration and the previous "best" or "fittest" points stored after the last 2 consecutive crunching phases of the BB–BC algorithm:

$$\overline{IV}_1 = \vec{P}(n) - \vec{P}(n-1), \tag{4}$$

where IV_1 stands for the improvement vector of single-step regression BB–BC, $P(n)$ is the current best or fittest point, and $P(n-1)$ is the last stored best or fittest point.

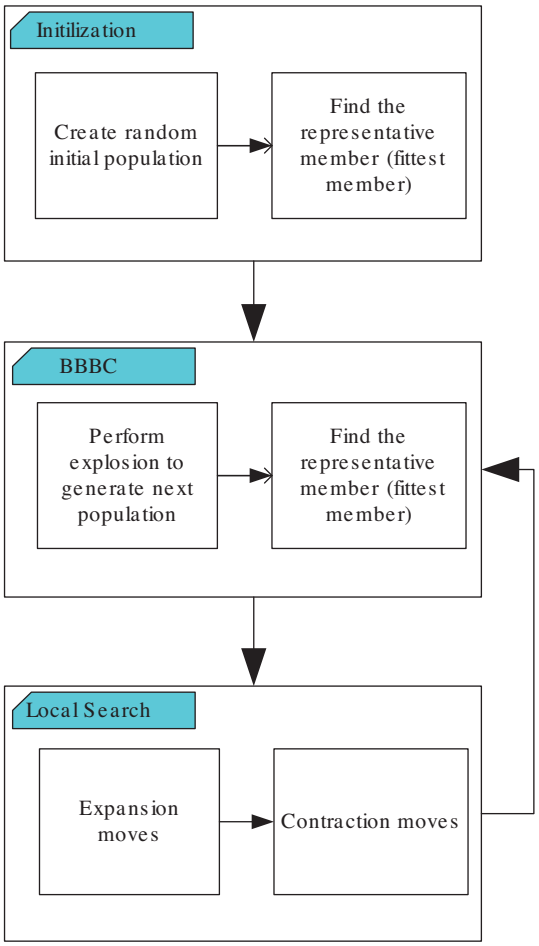

Figure 2. Generic algorithm flowchart.

In this version of the local search methodology, the memory usage is just for the single step; therefore, there is no information usage from the older crunching phase representative points. The flowchart of the local search part is given in Figure 3 and the search steps on the direction line are illustrated in Figure 4. The magnitude of the direction vector is halved after each unsuccessful expansion step. The number of halving operations (n_h) should be predetermined by the user. Only one contraction operation is allowed if all of the expansion trials turn out to be a failure. All of these predetermined parameters within these local move operations are not very firm constraints for the algorithm and they can be relaxed when needed with respect to the problem geometry.

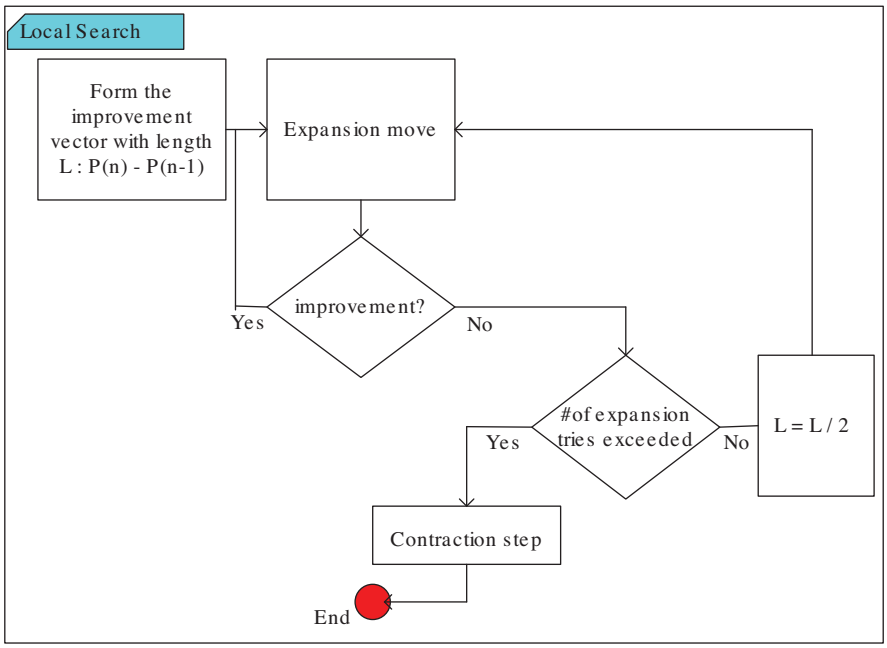

Figure 3. Local search phase for single-step regression in the BB–BC algorithm.

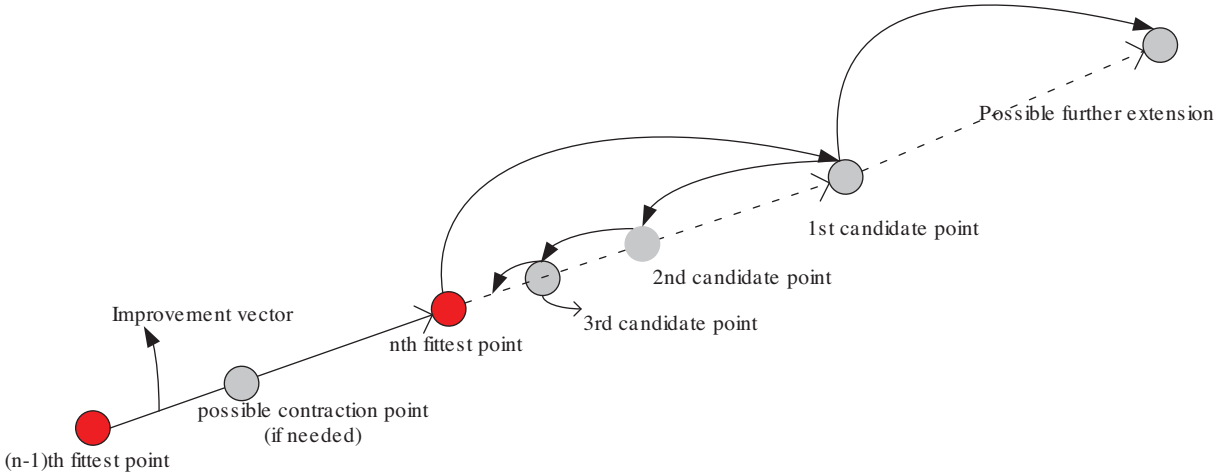

Figure 4. Illustration of the direction vector formation for the local improvement with single-step regression in the BB–BC algorithm.

4. Direction vector formation with double-step regression

The direction vector is the weighted mean of IV_1 and IV_2, where IV_2 is defined similar to IV_1:

$$\overline{IV}_2 = \vec{P}(n) - \vec{P}(n-2), \tag{5}$$

$$\overline{IV}_h = \alpha\overline{IV}_1 + (1-\alpha)\overline{IV}_2, \tag{6}$$

where α is a number in the interval [0, 1]. Note that if $\alpha = 1$, the double-step regression procedure reduces to single-step regression. There is information usage from both the $(n-1)$th and $(n-2)$th representative points; thus, this allows one to form a nonregular simplex for the local minimum search.

The bounds for the search direction are illustrated in Figure 5, where a possible direction vector is given for the case $\alpha = 0.5$.

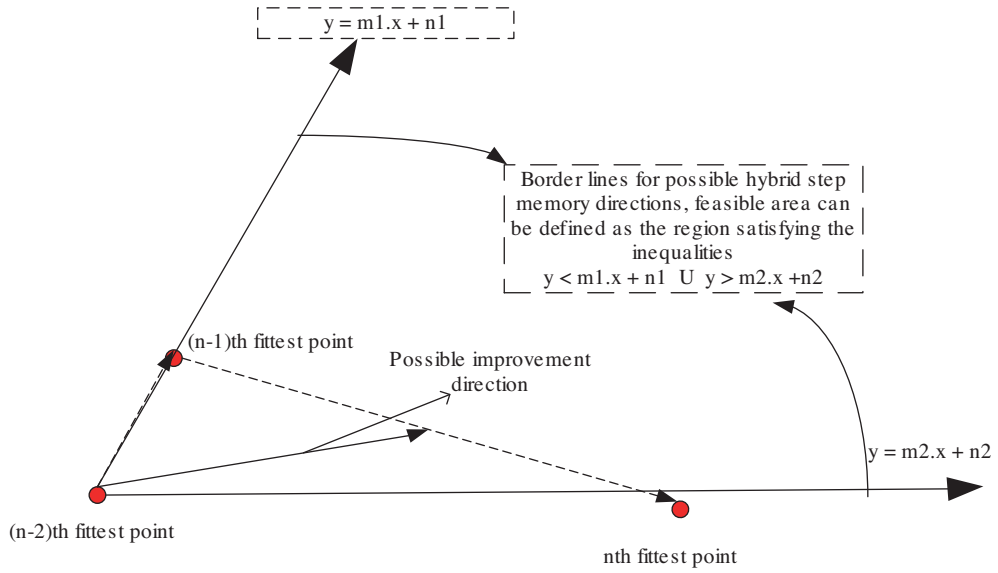

Figure 5. Illustration of the direction vector formation for the local improvement with double-step regression in the BB–BC algorithm.

5. Dichotomous search on local direction vector

A dichotomous search technique on the defined local direction vector can be utilized in the above mentioned local direction vector generation methods. The flowchart for the dichotomous search on the local direction vector for one-step regression in the BB–BC algorithm is illustrated in Figure 6.

The local search moves should fasten the movement for the global minima. The local search steps look for an improvement on the defined direction line but not necessarily a minimum point. Thus, the next explosion center of the bang phase is not guaranteed to be a local minimum. Moreover, the big bang phase of the BB–BC algorithm is still global in nature and these 2 factors avoid search stagnation.

Figure 7 serves as an illustration example of the representative point evolutions in applying both the original BB–BC algorithm and the BBBC–LS algorithm. In this example, the original BB–BC algorithm and the BBBC–LS has been run on the same objective function (Rosenbrock objective function: minimum at $(x = 1, y = 1)$, minimum cost $= 0$) with same parameters and same random number generator seeds and it starts from the same point $(x1 = 2.4721, y1 = 6.3589)$. In the following iterations, the fittest point was moved to another location, where it is shown as p in Figure 7. While the original BB–BC algorithm performs the next explosion centering on this point, the proposed hybrid algorithm replaces point p with p'. The same procedure follows for the whole run and the resulting trajectories are given in Figure 7. The hybrid BB–BC algorithm (BBBC–LS) clearly ends up at a point closer to the global minimum at $(1, 1)$, with a smoother trajectory. Note also that this simple example is given for the direction vector formation with the single-step regression case.

6. Simulation results

The simulation results are given in 2 subchapters. First, the effect of the local directional search over the original BB–BC is investigated. Next, the newly proposed method, including NM crunching, is compared with the state-

of-the-art methods on selected benchmark functions of the 2005 IEEE Congress on Evolutionary Computation Conference (CEC'05) real-parameter optimization session [17].

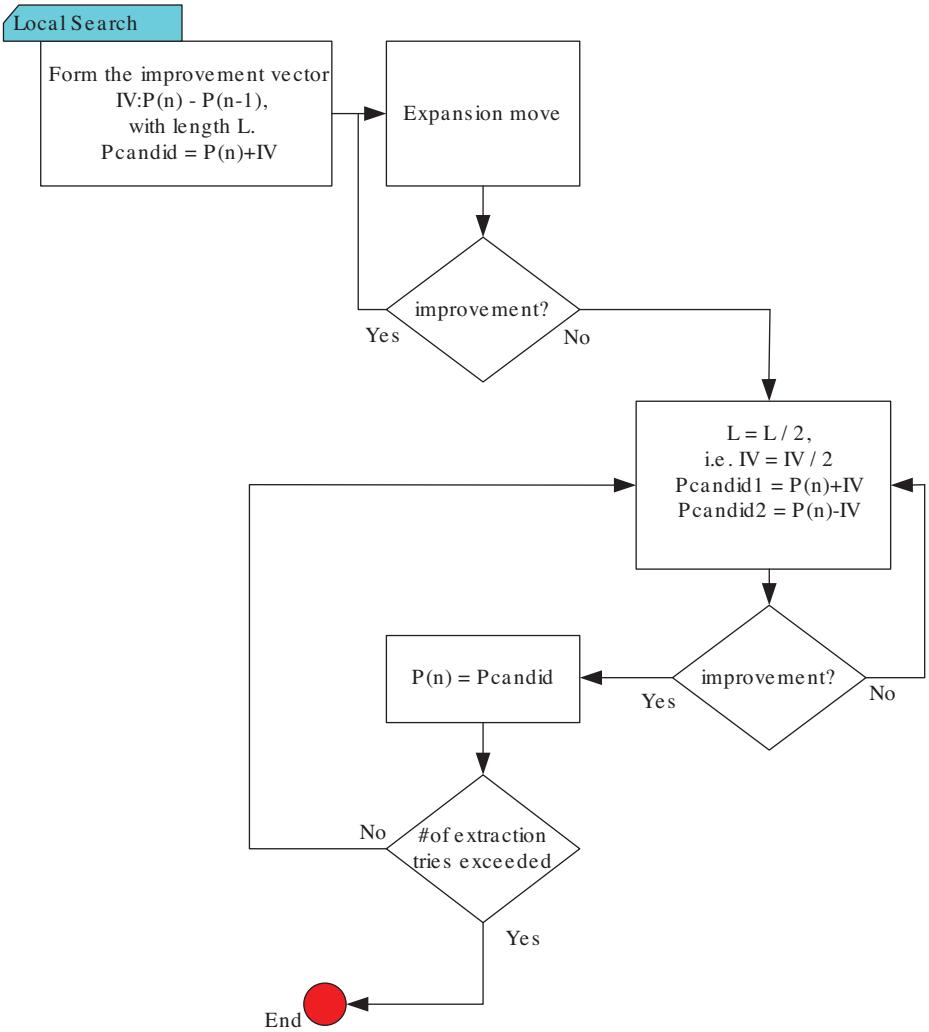

Figure 6. Flowchart for the dichotomous local search algorithm.

6.1. Inspecting the effect of local directional moves

The results of the hybrid method are compared with the results of the original BB–BC optimization algorithm on the test function suite, including the Rosenbrock, Rastrigin, Ackley, Sphere, Step, Ellipsoid, Griewank, and Schwefel functions. The first 6 test functions are chosen to be same as in the original paper presenting the BB–BC algorithm [1]; the final 2 functions are added to further enlighten the algorithm performance on the multimodal functions.

The stopping criterion is defined as the maximum number of fitness evaluations for both algorithms. If the stopping criterion had been chosen as the number of iterations, then the hybrid algorithm would have been advantageous compared to the original BB–BC algorithm, since the BBBC–LS searches for extra points around neighboring representative points after the crunching phase; therefore, the comparison would not have been fair. Moreover, in most of the practical problems, the main processing time is spent on the cost FE, though that is

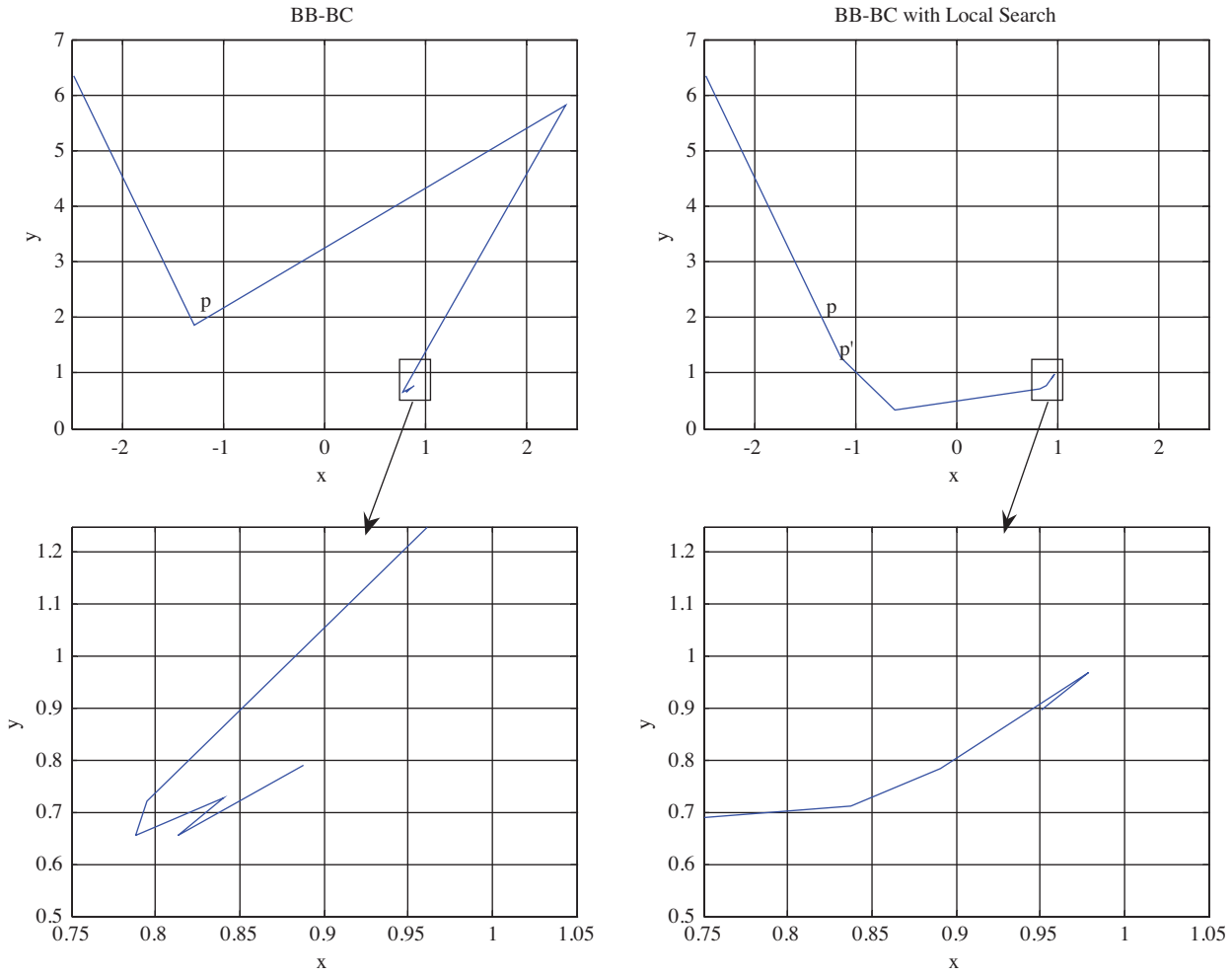

Figure 7. The hybrid BB–BC algorithm versus the original BB–BC algorithm: the upper left-hand side illustrates the movement of the best point under BB–BC; the upper right-hand side illustrates the movement of best point under BBBC-LS. The lower figures are the zoomed-in versions of the areas of interest.

not necessarily the case for our test functions. The time spent for the BB–BC and the BBBC–LS algorithms is almost the same; therefore, this criterion is not taken into account.

The results logged in this chapter are obtained from 10,000 random runs for each test. This number is more than enough for reliable statistical analyses. Tests are carried out for different population sizes (N), search space sizes, and numbers of FEs.

Tables 2 and 3 are arranged with respect to the search space size and the number of evaluations allowed. For each objective function, Table 2 gives the results of the search on a smaller space: [–10, 10] in both dimensions, while Table 3 reports the case where the search space is considerably large ([–50, 50] in both dimensions). For the Ellipsoid's step and sphere functions, the space topology is "easier" in comparison and the number of FEs (stopping criteria) is chosen to be half of that of the Ackley, Rastrigin, Rosenbrock, Griewank, or Schwefel counterparts. The number of halving, n_h, is selected as 7 in all of the test cases.

Figure 8 shows the convergence of the average best fitness of all of the runs summarized in Table 2. The convergence graphs related to Table 3 are omitted as they are similar.

Having more than one minimum (multimodality) causes the performance of excessive FEs around the local minima. Because of this, the performance improvement on the multimodal functions, Rastrigin and Griewank, are less than that of the unimodal functions. Inspecting Table 3 clearly reveals that as the search space is enlarged, the effect of the local directional moves increase. Moreover, even in the small search space, the hybrid method moves to the optimum point faster (Figure 8).

Table 2. Average costs for all of the functions. (N = 20, FE = 1000, search space: $[-10, 10]^2$).

Function	BB–BC	Single step regression (improvement %)	Double step regression (improvement %)	Dichotomous search (improvement %)
Ackley	0.65	0.49 (25%)	0.51 (22%)	0. 47 (28%)
Ellipsoid	0.11	0.07 (36%)	0.07 (36%)	0.06 (45%)
Rastrigin	1.35	1.15 (15%)	1.11 (18%)	1.18 (13%)
Rosenbrock	0.40	0.31 (23%)	0.30 (25%)	0.29 (28%)
Sphere	0.08	0.05 (38%)	0.05 (38%)	0.04 (50%)
Step	0.08	0.05 (38%)	0.05 (38%)	0.04 (50%)
Griewank	0.0143	0.0136 (5%)	0.0136 (5%)	0.013 (9%)
Schwefel	0.0019	0.0014 (26%)	0.0014 (26%)	0.0013 (32%)

Although the tables report only the average scores of the test runs, the median, standard deviation, and maximum deviation of the scores are quite consistent. They are omitted here in order not to diminish the readability of the tables.

One can easily conclude that a slight improvement in the local search step can cause further improvement in the total search performance by checking the second and last columns of the tables, that is, the results for the single-step regression and dichotomous search. Here, our purpose is not to intensify the local search step; the idea is to amalgamate the local and global search procedures. Better results could have been obtained by fine tuning the parameters of the local search algorithm, but the generality of the hybrid algorithm would have been sacrificed.

Table 3. Average costs for all of the functions. (N = 30, FE = 3000, search space: $[-50, 50]^2$).

Function	BB–BC	Single step regression (improvement %)	Double step regression (improvement %)	Dichotomous search (improvement %)
Ackley	1.14	0.87 (24%)	0.81(29%)	0.75 (34%)
Ellipsoid	0.33	0.19 (42%)	0.18 (45%)	0.16 (52%)
Rastrigin	1.96	1.63 (17%)	1.53 (22%)	1.65 (16%)
Rosenbrock	3.59	3.04 (15%)	3.20 (11%)	2.96 (18%)
Sphere	0.23	0.13 (43%)	0.12 (48%)	0.11 (52%)
Step	0.24	0.13 (46%)	0.12 (50%)	0.11 (54%)
Griewank	0.0097	0.0081 (17%)	0.0077 (21%)	0.0076 (22%)
Schwefel	0.0050	0.0030 (40%)	0.0029 (42%)	0.0027 (46%)

6.2. Comparison with the state-of-the-art algorithms

To evaluate the performance of the newly proposed hybrid method (BBBC–LS), the algorithm is applied to 4 test functions with distinct characteristics, selected from the benchmark test bed proposed for the CEC'05 Special

Session on Real-Parameter Optimization [17]. Three dimensional mappings for the 2-dimensional search spaces of the selected benchmark functions are given in Figures 9a–9d. In the simulations, 10-dimensional versions of these functions are used. The mathematical expressions, search range, global minimum function values, and basic properties for the benchmark functions are given in Table 4.

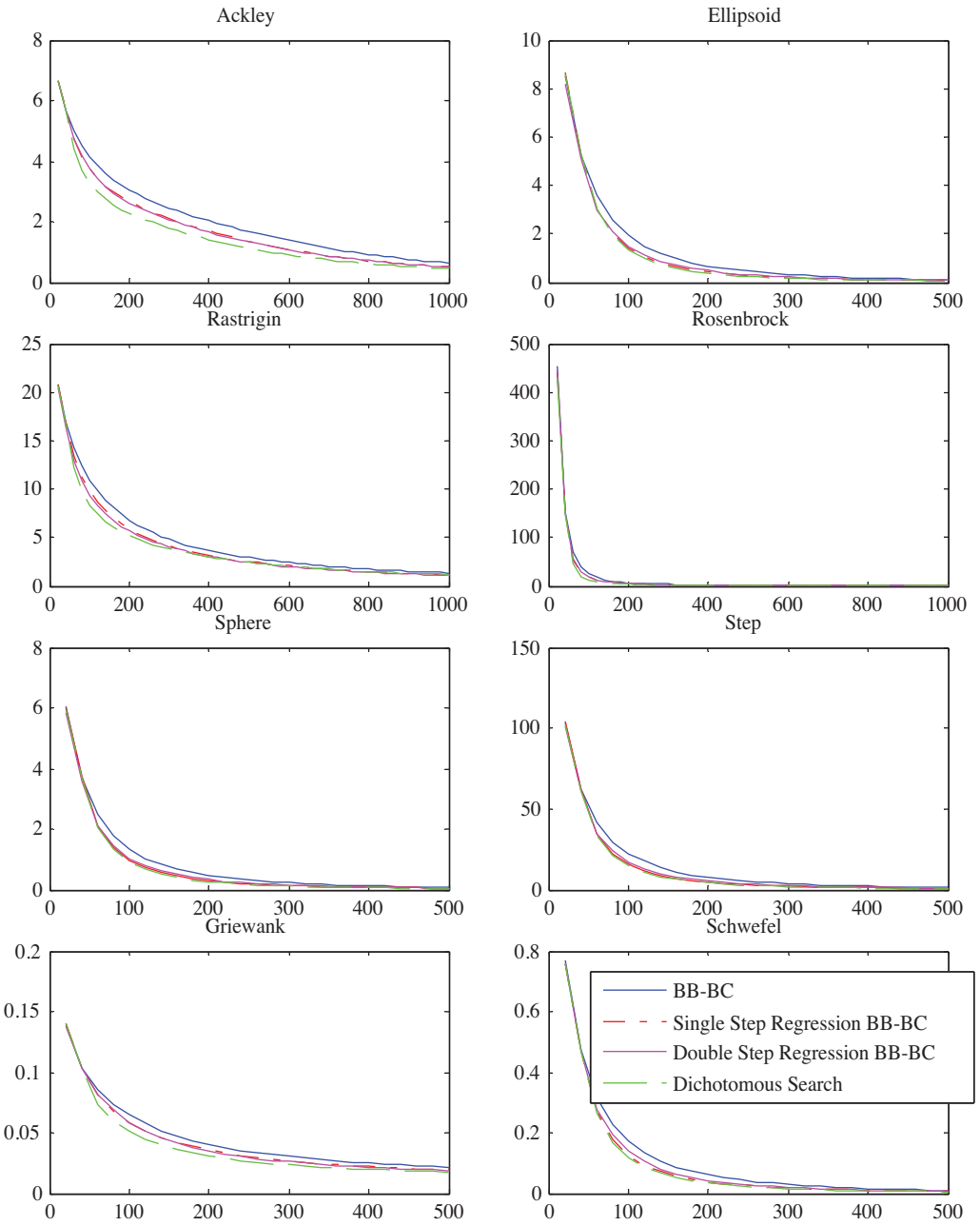

Figure 8. The improvements of the cost values for all of the functions for the small search space through the FEs.

To verify the effectiveness of the proposed approach, 3 well-known optimization routines are utilized on the same test functions: the genetic algorithm (GA) is probably the most commonly accepted umbrella term covering many variants. Covariance matrix adaptation evolutionary strategies (CMA–ES) and particle swarm

optimization (PSO) have also been successfully applied in many research and application areas over the past few decades. In this work, the GA is used as implemented in the Global Optimization Toolbox from MATLAB R2010a; CMA–ES are used as detailed in [18–20] and code is used as the January 2011 version in Hansen's web page [21]; and PSO Toolbox [22] is used for particle swarm evaluations.

Table 5 reports the benchmark function scores for all of the algorithms at the end of 500, 1000, and 2000 FEs for 1000 independent runs. In these simulations, the BBBC–LS algorithm uses the following parameters: $N = 15$, $n_h = 2$, $T_{fe} = 0.6$, $sm = 10$, and $nm_t = 0.1 \ / \ (1 + k / \ sm)$, where k is the iteration number. As the local directional move, a dichotomous search is used. The stopping criterion for the NM crunching phase is chosen to be the termination tolerance; therefore, the NM crunching budget (nm_b) parameter is set to infinity. The other algorithms are optimized only for the population size; all of the remaining algorithm-specific parameters are either left as the default/suggested parameters or self-tuned by the algorithm itself.

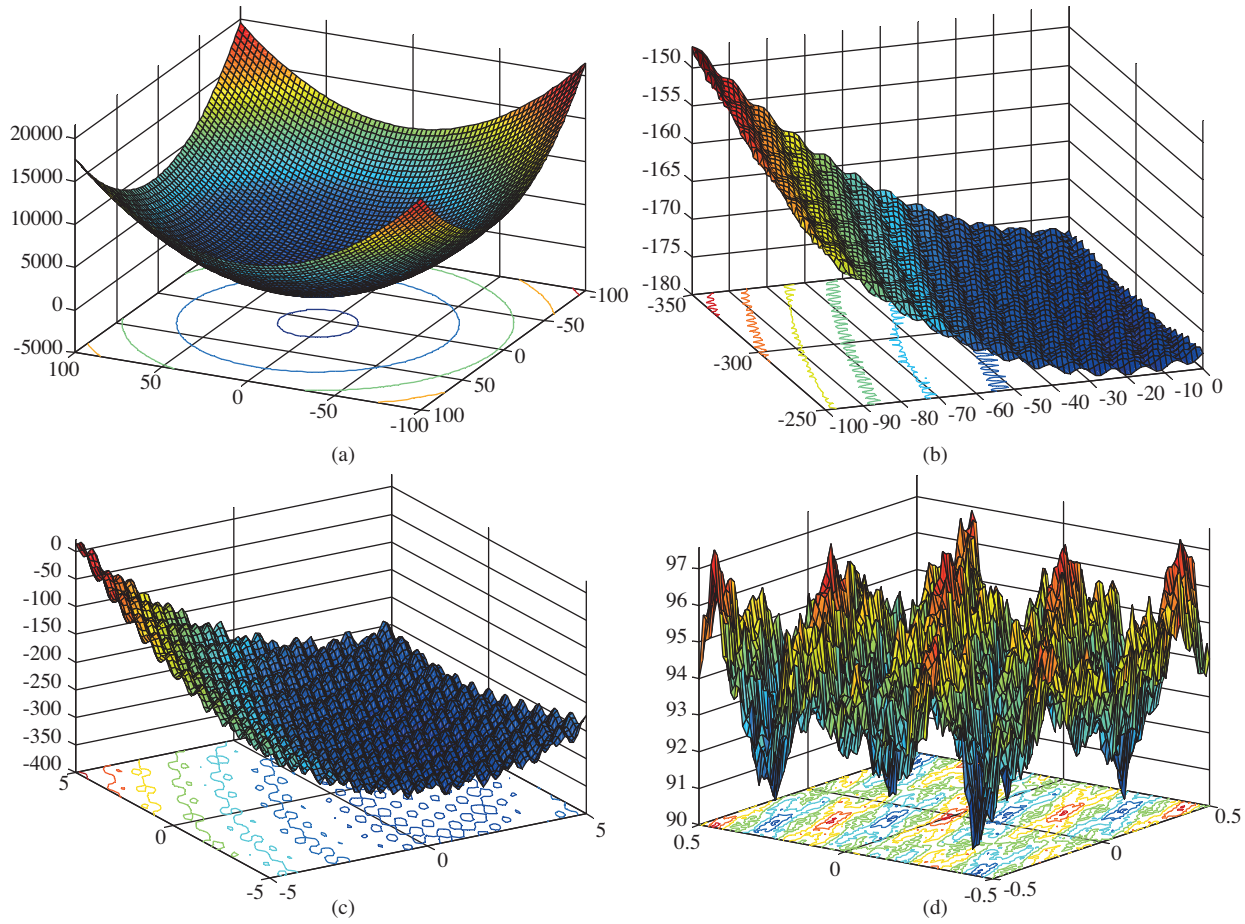

Figure 9. Benchmark test functions from the CEC'05 competition: a) shifted sphere, b) shifted rotated Griewank, c) shifted rotated Rastrigin, and d) shifted rotated Weierstrass.

Tables 6 and 7 serve for summarizing the performances of the algorithms. In Table 6, every entry gives the order for the corresponding algorithm at the end of the corresponding FE budget. Table 7 reports the best method on 4 test functions and 3 different FE levels (summing up 12 cases) and assigns an overall rating considering the mean place.

Table 4. Summary of the benchmark functions (D = 10).

Function	Min.	Search range	Properties
Shifted sphere: $F_{sphere} = \sum_{i=1}^{D} z_i^2 + bias_1$, $z = x - o$, $x = [x_1, x_2, ...x_D]$	$bias_1 = -450$	$x \in [-100, 100]^D$	Unimodal, shifted, separable, scalable
Shifted rotated Griewank: $F_{griewank} = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + bias_2$, $z = (x - o) * M$, $x = [x_1, x_2,....x_D]$, $M = M'(1 + 0.3 \lvert N(0, 1) \rvert)$, M': Linear Transformation Matrix with condition number 3	$bias_2 = -180$	$x \in [-500, 0]^D$	Multimodal, rotated, shifted, nonseparable, scalable
Shifted rotated Rastrigin: $F_{rastrigin} = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + bias_3$, $z = (x - o) * M$, $x = [x_1, x_2, ...x_D]$, M: Linear Transformation Matrix with condition number 2	$bias_3 = -330$	$x \in [-5, 5]^D$	Multimodal, rotated, shifted, nonseparable, scalable, lots of local optima
Shifted rotated Weierstrass: $F_{weierstrass} = \sum_{i=1}^{D} (\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (z_i + 0.5))]) - D\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k 0.5)] + bias_4$, $a=0.5$, $b=3$, $k_{max}=20$, $z = (x - o) * M$, $x = [x_1, x_2, ...x_D]$, M: Linear Transformation Matrix with condition number 5	$bias_4 = 90$	$x \in [-0.5, 0.5]^D$	Multimodal, rotated, shifted, nonseparable, scalable, continuous, differentiable only on a set of points

Table 5. Average performance scores.

Sphere	FE: 500	FE: 1000	FE: 2000	Griewank	FE: 500	FE: 1000	FE: 2000
GA	–433,502	–445,616	–448,277	GA	–53,481	–993,917	–116,699
CMA-ES	–449,688	–449,997	–450.000	CMA-ES	–169,161	–169,982	–170,983
PSO	–422,404	–441,127	–449,009	PSO	–167,573	–173,789	–179,555
BBBC-LS	**–448,667**	**–449,838**	**–449,979**	**BBBC-LS**	**–174,719**	**–177,160**	**–178,314**
Rastrigin	FE: 500	FE: 1000	FE: 2000	Weierstrass	FE: 500	FE: 1000	FE: 2000
GA	–231,448	–273,896	–276,315	GA	101,271	99,491	99,510
CMA-ES	–246,733	–274,883	–309,702	CMA-ES	101,640	97,047	94,414
PSO	–230,444	–262,460	–287,820	PSO	100,386	98,850	98,866
BBBC-LS	**–299,244**	**–309,962**	**–310,070**	**BBBC-LS**	**98,316**	**97,020**	**95,223**

The power of the BBBC–LS lies not only in its capability for quick convergence but also in its low level of complexity. There are a few parameters to be tuned: the user should select the population size (N), number of expansion/contraction steps (n_h), NM crunching budget (nm_b), NM crunching tolerance (nm_t), crunching phase switching parameter (T_{fe}), and the explosion strength adjusting parameter (sm). However, the GA, CMA-ES, and PSO have many parameters to be selected by the designer (Table 8), though many variants of these methods generally offer the self-selection/adaptation of these parameter settings.

Table 6. Order of the algorithms: 1) best, 2) second, 3) third, and 4) worst.

Sphere	FE: 500	FE: 1000	FE: 2000	Griewank	FE: 500	FE: 1000	FE: 2000
GA	3	3	4	GA	4	4	4
CMA–ES	1	1	1	CMA–ES	2	3	3
PSO	4	4	3	PSO	3	2	1
BBBC–LS	**2**	**2**	**2**	**BBBC–LS**	**1**	**1**	**2**
Rastrigin	FE: 500	FE: 1000	FE: 2000	Weierstrass	FE: 500	FE: 1000	FE: 2000
GA	3	3	4	GA	3	4	4
CMA–ES	2	2	2	CMA–ES	4	2	1
PSO	4	4	3	PSO	2	3	3
BBBC–LS	**1**	**1**	**1**	**BBBC–LS**	**1**	**1**	**2**

There are many metrics on algorithm complexity but neither of them is universally accepted. In CEC'05, the running time differences between 200,000 FEs ($T1$) and the complete computing time for the algorithm with 200,000 FEs ($T2$) have been normalized with a run time of a reference mathematical function ($T0$) on a dedicated computer. This can be formulated as:

$$Complexity = \frac{< T2 > -T1}{T0}, \tag{7}$$

where the $< . >$ symbols stand for the averaging function over multiple runs. The details of the complexity analysis can be further investigated in [17]. The results of the complexity analysis can be found in Table 8. The test function used for the complexity analysis is randomly chosen to be as the shifted rotated Weierstrass function.

Table 7. Summary of the algorithm comparisons.

Algorithm	# of first rankings	Average ranking	Overall rank
GA	0	3.5833	4
CMA-ES	4	2	2
PSO	1	3	3
BBBC-LS	**7**	**1.4167**	**1**

Table 8. Complexity analysis.

Algorithm	Complexity (CEC)	Run time (s)	# of parameters
GA	27.9551	189.6874	13
CMA-ES	44.9625	294.4760	24
PSO	25.1343	172.3075	14
BBBC-LS	**12.8308**	**96.5014**	**6**

7. Conclusion

A simple but effective hybridization procedure for the BB–BC optimization algorithm is presented. The method generates a direction vector from the past positions of the best individuals found so far and investigates this line with extraction or contraction moves. This local search phase is completely modular and works without interception into the original BB–BC algorithm. Moreover, the crunching phase of the algorithm is expanded to include a simplex-based approach, namely the NM optimization method.

The crunching phase using the NM optimization method improves the exploitation capability of the BB–BC algorithm, and so it is more appropriate to use it towards the final steps of the search. Therefore, the proposed method introduces a switching parameter (T_{fe}) for the crunching phase selection. Next, during the early iterations, the weighted mean of the candidate member solutions or the best solution member is selected as the center of mass, whereas after the switching condition is fulfilled, NM crunching is used for a more exploitive search. The switching threshold parameter is assigned at the beginning and kept constant throughout the search, but it is a promising idea to adapt this parameter in a dynamical manner. This adaptation could be performed based on a feedback controller observing the population diversity and history of the population diversity.

The simulation results on various test functions clearly illustrate the superiority of adding local directional moves over the original BB–BC algorithm. The accuracy achieved by the newly proposed method within the same number of fitness FEs is quite considerable and makes this routine worthy. Moreover, as a compact new algorithm, the BBBC–LS turns out to be a good alternative to the widely accepted state-of-the-art evolutionary optimization algorithms. Its accuracy is better or at least comparable for the tested benchmark functions and the complexity and running time are far better than the GA, CMA–ES, and PSO.

References

[1] O.K. Erol, İ. Eksin, "A new optimization method: big bang-big crunch", Advances in Engineering Software, Vol. 37, pp. 106–111, 2006.

[2] T. Kumbasar, E. Yeşil, İ. Eksin, M. Güzelkaya, "Inverse fuzzy model control with online adaptation via big bang-big crunch optimization", Proceedings of the 3rd International Symposium on Communications, Control, and Signal Processing, pp. 697–702, 2008.

[3] T. Kumbasar, İ. Eksin, M. Güzelkaya, E. Yeşil, "Big bang-big crunch optimization method based fuzzy model inversion", Lecture Notes in Computer Science, Vol. 5317, pp. 732–740, 2008.

[4] C.V. Camp, "Design of space trusses using big bang-big crunch optimization", Journal of Structural Engineering, Vol. 133, pp. 999–1008, 2007.

[5] A. Kaveh, S. Talatahari, "Size optimization of space trusses using big bang-big crunch algorithm", Computers and Structures, Vol. 87, pp. 1129–1140, 2009.

[6] H.M. Genç, O.K. Erol, İ. Eksin, "An application and solution to gate assignment problem for Atatürk Airport", Proceedings of the 6th IFAC International Workshop on Knowledge and Technology Transfer in/to Developing Countries: Automation and Infrastructure, pp. 125–130, 2009.

[7] M. Doğan, Y. Istefanopulos, "Optimal nonlinear controller design for flexible robot manipulators with adaptive internal model", IET Control Theory and Applications, Vol. 1, pp. 770–778, 2007.

[8] A. Akyol, Y. Yaslan, O.K. Erol, "A genetic programming classifier design approach for cell images", Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, pp. 878–888, 2007.

[9] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms", Caltech Concurrent Computation Program (report 826), 1989.

[10] N. Noman, H. Iba, "Accelerating differential evolution using an adaptive local search". IEEE Transactions on Evolutionary Computation, Vol. 12, pp. 107–125, 2008.

[11] M. Lozano, F. Herrera, N. Krasnogor, D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," Evolutionary Computation - Special issue on magnetic algorithms, Vol. 12, pp. 273–302, 2004.

[12] Y.S. Ong, A.J. Keane, "Meta-Lamarckian learning in memetic algorithms," IEEE Transactions on Evolutionary Computation, Vol. 8, pp. 99–110, 2004.

[13] Y.S. Ong, M.H. Lim, N. Zhu, K.W. Wong, "Classification of adaptive memetic algorithms: a comparative study", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 36, pp. 141–152, 2006.

[14] N.K. Bambha, S.S. Bhattacharyya, J. Teich, E. Zitzler, "Systematic integration of parameterized local search into evolutionary algorithms," IEEE Transactions on Evolutionary Computation, Vol. 8, pp. 137–155, 2004.

[15] C.W. Ahn, E. Kim, H.T. Kim, D.H. Lim, J. An, "A hybrid multiobjective evolutionary algorithm: striking balance with local search", Mathematical and Computer Modeling, Vol. 52, pp. 2048–2059, 2010.

[16] T.A. El-Mihoub, A.A. Hopgood, L. Nolle, A. Battersby, "Hybrid genetic algorithms: a review", Electronic Letters, Vol. 13, 2004.

[17] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization", Tech. Report, Nanyang Technological University, 2005.

[18] N. Hansen, S.D. Müller, P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA–ES)", Evolutionary Computation, Vol. 11, pp. 1–18, 2003.

[19] N. Hansen, The CMA evolution strategy: a comparing review, In: J.A. Lozano, P. Larrañga, I. Inza, E. Bengoetxea (eds.), Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms, New York, Springer, pp. 75–102, 2006.

[20] N. Hansen, A.S.P. Niederberger, L. Guzzella, P. Koumoutsakos, "A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion", IEEE Transactions on Evolutionary Computation, Vol. 13, pp. 180–197, 2009.

[21] N. Hansen, "Covariance matrix adaptation evolutionary strategy", Available at: http://www.lri.fr/∼hansen/.

[22] G. Evers, "Matlab particle swarm optimization research toolbox", Available at: http://www.georgeevers.org/pso_research_toolbox.htm.