Coordinate Exhaustive Search Initialization for Big Bang Big Crunch Optimization Algorithm

Osman Kaan EROL[1], Ibrahim EKSIN[2]

[1,2] Istanbul Technical University, Control and Automation Engineering Dept. Turkey
okerol@itu.edu.tr, eksin@itu.edu.tr

Abstract

Optimization Algorithms, especially evolutionary algorithms, have gained wide acceptance among many disciplines such as electrical, control or industrial engineering. The ability to solve an objective or cost function with more unknown parameters than known equations, which make the problem unsolvable by means of deterministic approaches, is the main benefit of using evolutionary algorithms. In all cases optimization algorithms rely on starting with a completely random initial solution set then evolves this set towards better ones in respect to fitness or objective function iteratively. In this study, we have proven that starting from a unique point after a brief local and deterministic search instead of a pure random set is more beneficial in respect to fitness function evaluation count, or computation time. This approach, although it can be applied to any optimization algorithm, is a natural add-on to Big Bang – Big Crunch (BBBC) optimization method.

1.Introduction

The evolutionary optimization algorithms can be broadly classified as synthetically invented ones [1-2] or nature inspired complex algorithms [3-8]. Every algorithm, being either synthetically invented or nature inspired, are expected to find one unique solution namely global optimal point, a number or parameter that minimizes or maximizes a given function called objective/cost/fitness function, without getting trapped with surrounding local optima. It is a well-known fact that there is no optimization method to cope with every kind of objective functions with highest performance. Each year, Congress on Evolutionary Computation [9] is organized where many optimization algorithms are tested on a number of shifted, rotated functions which have many local optima.

In the literature, there is increasing number of global optimization methods. A common feature of these algorithms is that they use some form of probabilistic variable inclusion. Whether called crossover or mutation in Genetic Algorithms (GAs) [4], or heating in simulated Annealing [8], without using randomness, the algorithms may get stuck to local optima. The inclusion of randomness oppose to quick convergence induced by local search part of the algorithms, however guarantees the convergence to global optimum point.

Optimization algorithms are used by engineers from different disciplines therefore the algorithm must be simple to be used with less adjusting parameters, so does the Big Bang Big Crunch (BBBC) algorithm. With a single adjusting parameter, which is the size of the universe, BBBC is simple to use yet perform better than many state-of-art algorithms [10].

Devaluating the general nature of algorithm and injecting some problem specific features into the optimization method will have a doping effect on these specific kinds of problems. For example, the initial members can be generated according to an a priori knowledge. If the optimal point is supposed to be bounded within a limited region of the entire search space, then new generation's members are tried to be located within this region either at the initialization stage or after the application of mutation operator [11-13]. The more this adaptation is specific to the problem, the more success can be expected in solving these problems. Adding a new feature that shifts the algorithm from general to a specific one is cumbersome if one still wants to preserve the generality.

In this study we conduct first an exhaustive search over the coordinate axis in an effort to collect a meaningful knowledge on the nature of the problem. A similar methodology called "cyclic coordinate search" is done in [14] as a part of a deterministic search strategy. Since global optimization methods rely on the probabilistic functions, without getting sacrifice on the global property of the methodology given in this study, we have adopted and modified the idea of the coordinate search and used this prior to BBBC method. The initialization has performed so well that in most cases, the exact solution is found during the initialization stage.

In Section 2, a brief introduction to BBBC algorithm is given while in Section 3, new algorithm is covered in detail and finally in Section 4, simulation results are given and concluded.

2. BBBC Algorithm

BBBC optimization method [15] is inspired from the Big Bang and Big Crunch theories of the universe. In Big Bang phase, as of the theory implies, from a unique point (called center point or center of mass) a whole new universe is being populated. When the algorithm starts from initial, since the localization of this center point is not known, the population is spread over the entire universe (the search space) with equal probability only once. New population is formed by using the Gaussian distribution function as given in Eq.1

$$x_i = x_c + ex \times randn \tag{1}$$

Here xi stands for an element of the new population. The term ex denotes the "explosion strength" while the term "randn" is the randomization function used in MATLAB. In Eq.1, it is expected that the population members will be accumulated around the center point xc. This can be viewed as diversification around xc. However, if the localization of an optimal point can be guessed a priori, then a specific center point of the initial population can be used in Eq.1 which is the motivation behind this study.

The crunching step is done by using the weighted average operation such as given in Eq.2

$$x_c = \frac{\sum f_i x_i}{\sum f_i} \tag{2}$$

Here the weights of the candidate points are their respective fitness measures labeled as fi in Eq.2.

The successive calling of the banging and crunching functions will conduct a thorough search to find the optimal point. The population found after the banging function will be the input population for the crunching function and vice versa. One banging function used together with one crunching function form an iteration of the search. The search can be forced to localize around the center point by decreasing the diversification factor. This can be done by the insertion of the term "iteration number" into Eq. 1 such as given in Eq.3. In this case as the iterations elapse, the algorithm will intensify its search around a specific point at the expense of spending more time around this point if the optimal point is different than this point.

$$x_i = x_c + \frac{ex \times randn}{iteration\_number} \tag{3}$$

3. BBBC Algorithm with Coordinate Exhaustive Search Initialization

Coordinate exhaustive search lies on the principle of conducting an exhaustive search over the coordinate axis only by changing the value of one parameter over all its range with a predefined sufficiently small step while keeping all the other parameter's values fixed to zero. The parameter which gives the minimum value of the objective function for the selected coordinate is recorded and the algorithm switches to the next coordinate again with all the other parameters kept to zero. This procedure is repeated until all parameters' coordinates are exhausted and minimum values for each recursion have been found and recorded. At the end of the algorithm, a unique point is formed by using the recorded parameter values hoping to find a reasonably good starting point. Therefore, this procedure differs from its predecessor cyclic exhaustive search in respect to keeping the value of the optimum parameter fixed in the next cycle of the search without an update procedure. Thus, an optimal candidate initial point has been formed at the end of the search by taking minimum of all so as to form a unique point. The procedure can be given with a flowchart as in Fig.1. As it is seen from the Fig.1, the search preferences have been reversed in this study. Almost all of the evolutionary computation techniques start with a highly diversified phase then switch to a local search after a certain convergence to the solution point, which may be called intensification phase. In contrast to this conventional or routine approach we start by conducting an exhaustive search in a very gross manner then switch and begin with the evolutionary algorithm.

Even though any evolutionary algorithm may actually use this reverse approach there exist a natural conformity between this initialization stage and the BB-BC evolutionary algorithm by itself since the BB-BC algorithm makes its start from a unique point, namely "center of mass point" between two successive iterations.

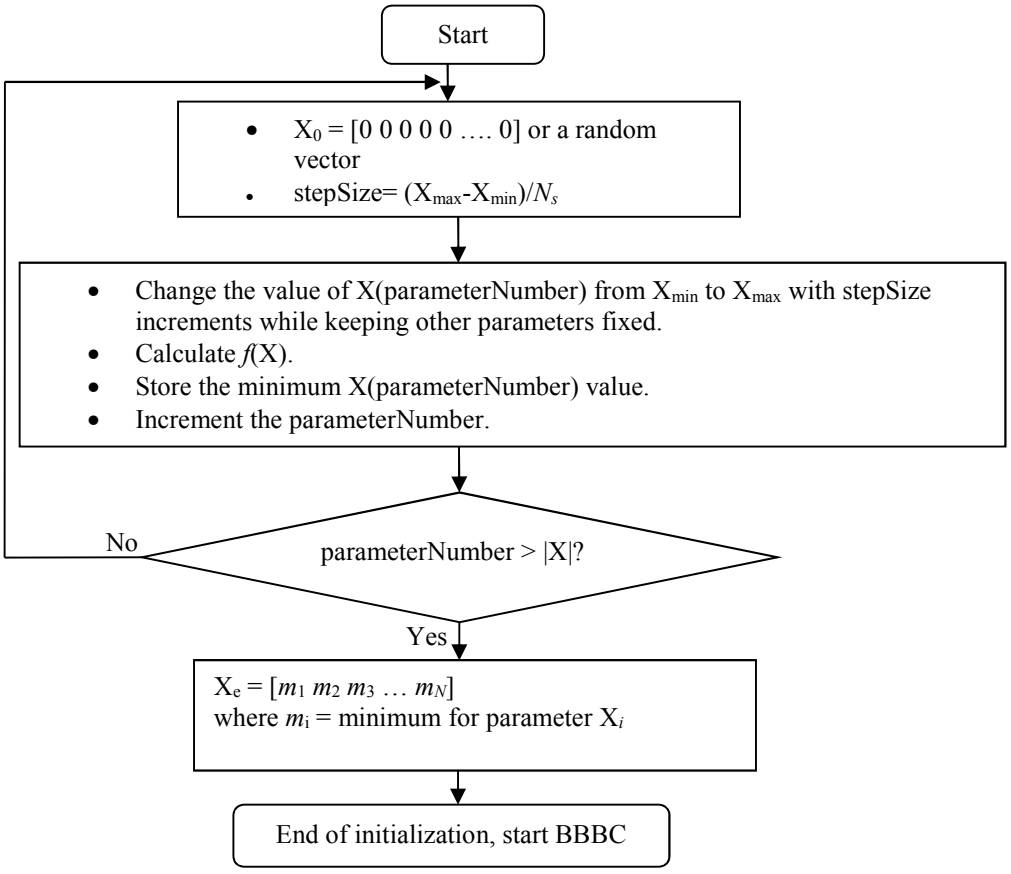

Fig.1 The flowchart of coordinate exhaustive search algorithm

That is, if GA, PSO or any other evolutionary algorithm is to be used then that valuable initial point so obtained can well be injected in the initial population with a predefined percentage. Graphical illustrations of the coordinate and random coordinate search methodologies are given if Fig.2 (a) and (b) respectively. Random coordinate search is only for simulating function shift effect.

4. The Experimental Results

The BB-BC algorithm with exhaustive coordinate search initialization has been tested on 34 different test functions with distinct characteristics, which are selected from the benchmark test bed proposed for the CEC'05 Special Session on Real-Parameter Optimization [10]. The complete list of these functions with their dimension and search space ranges is given in Table 1.

Table 1. List of numerical benchmark functions

Code	Function name	Dimension	Search Space
f1	Sphere function	30	[-5.12, 5.12]
f2	Axis parallel hyper-ellipsoid	30	[-5.12, 5.12]
f3	Schwefel's problem 1.2	20	[-65, 65]
f4	Rosenbrock's valley	30	[-2, 2]
f5	Rastrigin's function	10	[-5.12, 5.12]
f6	Griewangk's function	30	[-600, 600]
f7	Sum of different power	30	[-1, 1]
f8	Ackley's path function	30	[-32, 32]
f9	Beale function	2	[-4.5, 4.5]
f10	Colville function	4	[-10, 10]
f11	Easom function	2	[-100, 100]
f12	Hartmann function 1	3	[0, 1]
f13	Hartmann function 2	6	[0, 1]
f14	Six Hump Camel back function	2	[-5, 5]
f15	Levy function	30	[-10, 10]
f16	Matyas function	100	[-10, 10]
f17	Perm function	4	[-4, 4]
f18	Michalewicz function	10	[0, pi]
f19	Zakharov function	30	[-5, 10]
f20	Branins's function	2	[-5, 15]
f21	Schwefel's problem 2.22	30	[-10, 10]
f22	Schwefel's problem 2.21	30	[-100, 100]
f23	Step function	30	[-100, 100]
f24	Quartic function	30	[-1.28, 1.28]
f25	Kowalik's function	4	[-5, 5]
f26	Shekel's Family	4	[0, 10]
f27	Shekel's Family	4	[0, 10]
f28	Shekel's Family	4	[0, 10]
f29	Tripod function	2	[-100, 100]
f30	De Jong's fnc. 4 (no noise)	2	[-1.28, 1.28]
f31	Alpine function	30	[-10, 10]
f32	Schaffer's function 6	2	[-10, 10]
f33	Pathological function	5	[-100, 100]
f34	Inverted cosine wave function	5	[-5, 5]

The algorithm has been tested on 100 independent runs with 15 individuals within each generation. If the evolutionary part reaches the above mentioned values before 1000000 objective function evaluation, then this is considered as "one hit". At the end of the independent runs, the number of function calls has been averaged. If the evolutionary BB-BC algorithm part has used up all of its resources; that is, the top number of 1000000 function calls has been exhausted then it is considered as a "fail case". The result of the simulations is given in Table 2. It is obvious that smaller values indicate a faster convergence, hence success of the related algorithm. In this respect, the smallest value shows which algorithm is the winner for every test case. The number of winning cases and total function evaluation count are summed up to obtain a global measure of the performance of the methods compared at the bottom line of Table 2. If all the algorithms use 1000000 iterations and they have not reached the "Value to Reach" or VTR value, then no winner is assigned to that test case.

In Table 2, the results of CES-BBBC algorithm are obtained with respect to three different values for Ns, namely 100, 10 and 4. When the winning cases and the number of function calls are considered together small Ns values are more adequate for high dimensional functions such as functions f1-f8, f15, f16, f22-f24, f31. It can also be deduced from the results given in Table 2 that high Ns values should only be used in the case of low dimensional functions in a small interval of search domain such as functions f14, f30 to get better resolution right from the start of the algorithm. The table also point out that medium Ns value such as 10 should be used for functions which has dimensionality of greater than 4. In any case, even if CES-BBBC-4 has been used alone, then the method will record 15 winning cases over classical BBBC and Opposition based BBBC. Therefore, it would be wiser to use small number of search points such as 4 for all functions.

In the second part of the study, instead of conducting a search over the standard coordinate axis, search is conducted on a randomly chosen axis to artificial shift effect on the fitness function. In this case, instead of fixing all the constant parameters to zero, they are kept fixed at a given random point which changes with respect to the position of coordinate as given in Fig.1. The results of the newly introduced Random Coordinate Exhaustive Search BBBC (RCES-BBBC) method for various Ns values are given in Table 3.

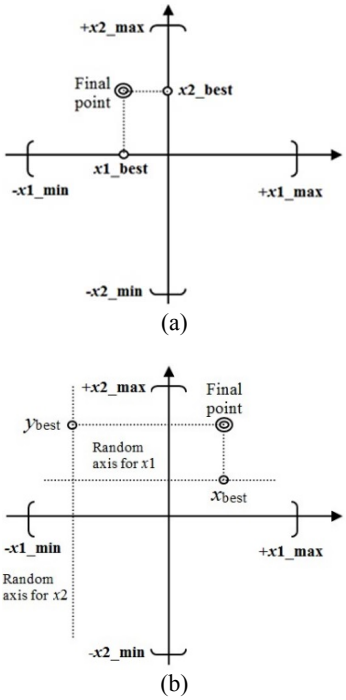

Fig.2 Illustration of (a) Coordinate Exhaustive Search (CES), and (b) Random Coordinate Exhaustive Search (RCES) in R^2

Table 2. Comparison of the algorithms

Function Code	Average Number of Function Calls			
	BBBC	**CES-BBBC-100**	**CES-BBBC-10**	**CES-BBBC-4**
f1	11980	3015	315	**135**
f2	42100	3015	315	**135**
f3	104330	2015	215	**95**
f4	861840	840600	849360	**840453**
f5	1000000	1015	115	**55**
f6	267330	3015	315	**135**
f7	639	3015	315	**135**
f8	734320	3015	315	**135**
f9	160210	130420	100160	**179**
f10	11796	11520	11554	12841
f11	1000000	1000200	1000020	1000008
f12	1000000	1000300	1000030	1000012
f13	1000000	1000600	1000060	1000024
f14	305,1	**215**	279,5	341
f15	1000000	**3015**	127696	24717
f16	**107,7**	10015	1005	415
f17	236730	257530	**106150**	303600
f18	1000000	1001000	1001000	1000040
f19	**24991**	28078	25893	25097
f20	**60194**	140390	400150	200203
f21	292560	3015	315	**135**
f22	237530	3015	315	**135**
f23	62611	3015	315	**135**
f24	**2620,8**	5522,3	2919	2721
f25	81,45	415	104,5	**77,5**
f26	1000000	1000400	1000400	1000016
f27	1000000	1000400	1000400	1000016
f28	1000000	1000400	1000400	1000016
f29	602110	**532500**	601244	801220
f30	**17,7**	215	35	23
f31	1000000	3015	315	**135**
f32	73,05	215	35	**23**
f33	636290	515	65	**35**
f34	1000000	1000500	1000500	1000020
Total of Function Evaluations	15350766,8	**9995130,3**	10232625	10213462,5
Number of Wins	5	3	1	**16**

Table 3. Comparison of the RCES-BBBC and previous form of BB-BC algorithms

Function Code	Average Number of Function Calls	
	BBBC	**RCES-BBBC**
f1	11980	**3015**
f2	42100	**3015**
f3	104330	103940
f4	861840	**853480**
f5	1000000	**1015**
f6	267330	**3015**
f7	**639**	3015
f8	734320	**3015**
f9	160210	**110410**
f10	11796	**10866**
f11	1000000	1000200
f12	1000000	1000300
f13	1000000	1000600
f14	**305,1**	473,45
f15	1000000	**3015**
f16	**107,7**	10128
f17	236730	126530
f18	1000000	1001000
f19	**24991**	27274
f20	**60194**	130370
f21	292560	**3015**
f22	237530	493470
f23	62611	**3015**
f24	**2620,8**	5483,4
f25	**81,45**	466,6
f26	1000000	1000400
f27	1000000	1000400
f28	1000000	1000400
f29	602110	721430
f30	**17,7**	215
f31	1000000	**3015**
f32	**73,05**	297,2
f33	**636290**	655740
f34	1000000	1000500
Total # of Function Evaluations	15350766,8	**11282523,65**
Number of Wins	10	**12**

Once again, both in terms of function evaluation counts and the number of winning cases, the RCES-BBBC with Ns value of 100 is the winner with a total of 11282523.65 and 12 cases respectively.

5. Conclusion

Almost all of the evolutionary algorithms use "first diversification and then intensification routine" approach; Here, we propose a paradigm shift by introducing a reversed mode to this conventional approach. In other words, we first execute a rough "intensification" or a local optimization algorithm to get a priori information about the function and then switch to the basic "diversification" or global evolutionary search phase.

In that respect, a primitive local search algorithm CES is adopted and modified as an initial step of the plain BB-BC evolutionary optimization method without sacrificing the global property of BB-BC methodology. The plain BB-BC optimization method combined and enhanced with three versions of coordinate exhaustive search initialization is tested on a vast number of test functions with distinct characteristics. The overall number of function calls decreased about 50 percent on the average when compared to the plain BB-BC optimization algorithm. As a consequence, it can be quite easily concluded that the speed of convergence of the newly proposed method has been ameliorated beyond comparison with respect to the plain BB-BC optimization algorithm. As a final comment, this reverse mode approach can easily be adapted and applied to all of evolutionary optimization algorithms with some slight modifications.

6. References

[1] R. Hooke, T.A. Jeeves, T.A. "Direct search" solution of numerical and statistical problems". Journal of the Association for Computing Machinery (ACM) 8 (2): 212–229, 1961

[2] Nelder, J.A., Mead R. "A Simplex Method for Function Minimization". Computer Journal 7: 308–313, 1965

[3] Holland JH. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: The University of Michigan Press, 1975

[4] Goldberg D.E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989

[5] Ong Y.-S., Lim M.-H., Zhu N., and Wong K.-W., "Classification of Adaptive Memetic Algorithms: A Comparative Study," IEEE Trans. Syst., Man, Cybernetics-Part B, vol. 36(1): 141–152, 2006

[6] Bambha N. K., Bhattacharyya S. S., Teich J., and Zitzler E., "Systematic integration of parameterized local search into evolutionary algorithms," IEEE Trans. Evolutionary Computation, vol. 8(2): 137–155, 2004

[7] Dorigo M, Birattari M, Stutzle T. Ant Colony Optimization. IEEE Computational Intelligence Magazine;1(4): 28-39, 2006

[8] Kirkpatrick S., Gelatt C. D., Vecchi M. P. "Optimization by Simulated Annealing". Science 220 (4598): 671–680, 1983

[9] Suganthan P.N., Hansen N., Liang J.J., Deb K., Chen Y.P., Auger A., Tiwari S., "Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real Parameter Optimization", Tech. Report, Nanyang Technological University, 2005

[10] Genc H.M., Eksin I., Erol O.K., "Big bang-big crunch optimization algorithm with local directional moves", Turk. J. Elec. Eng. & Comp. Sci., vol.21(5): 1359-1375, 2013

[11] Chou C.H. and Chen J.N. "Genetic algorithms initialization schemes and genes extraction," Proceeding of the 9th IEEE International Conference on Fuzzy Systems, vol. 2: 965-968, 2000

[12] Burke E.K., Newall J.P. and Weare R.F. "Initialization Strategies and Diversity in Evolutionary Timetabling", Evolutionary Computation Journal, vol 6.1: 81-103, 1998.

[13] Cao Y. and Wu Q. H. "Study of Initial Ppopulation in Evolutionary Programming", Proceedings of the European Control Conference, 1997

[14] Luenberger D.G., Linear and Nonlinear Programming. Reading, MA:Addison-Wesley, 1984

[15] Erol O.K., Eksin I., "A New Optimization Method: Big-Bang Big-Crunch", Advances in Engineering Software, Elsevier, vol 37(2): 106-111, 2006