MATEC Web of Conferences **73**, 04012 (2016) DOI: 10.1051/matecconf/20167304012
TPACEE-2016

Multi-objective Truss Optimization Using Different Types of the BB-BC Algorithm

Aleksandar Milajić [1,*] and *Dejan* Beljaković [1]

[1]University Union Nikola Tesla, 11000 Cara Dušana 62-64, Belgrade, Serbia

Abstract. Optimum design of truss structures is considered as a benchmark problem in the field of the structural optimization. In order to solve this hard combinatorial problem, it is necessary to implement adequate optimization tool that would provide sufficiently wide range of possible solutions within a reasonable time as well as to obtain good exploration and exploitation of search space. The aim of presented study was to compare efficiency of different multi-objective algorithms in solving this task.

1 Introduction

Size optimization of truss structures involves determining optimum values for member cross-sectional areas that would minimize the weight of a given truss structure while satisfying a number of inequality constraints that limit design variable sizes and structural responses. However, minimum weight is often not the only goal, because limited deflections and deformations are often more important demand in order to achieve usability demands in exploitation during the structure's life-cycle. These cases belong to the field of multi-objective optimization.

The main task of every optimization tool is to perform effective and efficient exploration of a given search space. This search for an optimal solution should be versatile enough to both intensively explore promising areas of the search space around high quality solutions, and to reach its unexplored areas. Two basic concepts for reaching these goals are usually called intensification and diversification. These terms originate from the tabu search field, but same or related concepts can also be found in other methods, such as evolutionary algorithms, denoted as exploitation (related to intensification) and exploration (related to diversification). The main difference between intensification and diversification is that intensification feature of a given search tool focuses on examining neighbors of elite solutions, while the diversification encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before [1]. A meta-heuristic will be successful on a given optimization problem if it can provide a good balance between the exploitation of the accumulated search experience and the exploration of the search space to identify regions with high quality solutions in a problem specific, near optimal way [2].

* Corresponding author: aleksandar.milajic@gmail.com

In the last decades, different nature inspired evolutionary algorithms have been developed and employed for structural optimization. These algorithms do not require for a given function to be derivable and an explicit relationship between the objective function and constraints is not needed. Big Bang – Big Crunch (BB-BC) algorithm, introduced by Erol and Eksin [3], is relatively new optimization method that relies on one of the theories of the evolution of the universe namely, the Big Bang and Big Crunch theory and has a low computational time and high convergence speed. The aim of presented research was to explore applicability of different variants of basic and hybrid multi-objective BB-BC in order to find an efficient and effective optimization tool for optimum design of a given truss structure that would be able to deal with numerous variables and to provide good solutions. Obtained results confirmed that the hybrid versions of the BB-BC algorithm outperform the basic one in exploration and exploitation of a search space for remarkably less time, as well as in producing much better Pareto fronts of possible solutions. Analysis of the results shows that hybridizing basic BB-BC with other meta-heuristic or some of their features can further improve its performance in solving hard combinatorial problems such as optimum structural design of truss structures.

2 Problem formulation

Mathematically, the optimal design of a truss can be formulated as finding set of variables $[A_1, A_2, ..., A_n]$, $A_i \in D$, where A_i is the cross-sectional area of a member i, n is the number of members in a givent russ structure, and D denotes the allowable set of values for the design variable A_i, in order to minimize both the nodal displacements (1) and the total weight of structure (2):

$$\delta_j = \delta_{j,\min}, j = 1,2,...,n_n \tag{1}$$

$$\mathrm{W(A)} = \sum_{i=1}^{n} \gamma_i A_i L_i \tag{2}$$

where n_n is nuber of nodes; $W(A)$ is weight of the structure; n is the number of members of the structure; γ_i represents the material density of member i and L_i is the length of member i, subject to constraints:

$$g_j(A) \le 0, j = 1,2,...,n \tag{3}$$

Since the presented problem has two objective functions, an appropriate solving method is the multi-objective tool that would be able to locate multiple Pareto optimal solutions in a single run. A solution is said to be Pareto optimal if and only if it is not dominated by any other solution in the performance space. If one solution dominates another, it implies that the first one is non-inferior to the second one for all the considered performance criteria but it is better than it for at least one criterion. All Pareto solutions form a Pareto front in the performance space.

Analysis of the whole Pareto front provides useful information on trade-off relationship between the fitness functions and enables a decision maker to consider different alternatives and make a choice that would represent acceptable compromise for conflicting objectives. In hard combinatorial problems such as this one it is impossible to conduct thorough search for Pareto solutions within the whole search space without appropriate optimization tool.

3 Optimization Method

Problem defined by the fitness functions (1) and (2) and set of constrains (3) is a multi-objective optimization problem with conflicting criteria. Consequently, there is no one optimal solution but a whole set of acceptable solutions called Pareto solutions or Pareto front. A solution is Pareto optimal if and only if it is not dominated by any other solution in the search space. Analysis of the whole Pareto front provides useful information on trade-off relationship between the fitness functions and enables a decision maker to consider different alternatives and make a choice that would represent acceptable compromise for conflicting objectives. In hard combinatorial problems such as this one it is impossible to conduct thorough search for Pareto solutions within the whole search space without appropriate optimization tool.

The BB-BC algorithm is relatively new evolution algorithm introduced in 2006 by Erol and Eksin [3]. The BB–BC method is inspired by a theory of the evolution of the universe, namely the Big Bang and Big Crunch theory. Every step of the algorithm consists of two phases: a Big Bang phase, and a Big Crunch phase. Similar to other evolutionary algorithms, initial solutions in the first Big Bang are created randomly and spread all over the search space in a uniform manner. The Big Crunch is a convergence operator that has many inputs but only one output, named as the *centre of mass*, where the term *mass* refers to the inverse of the fitness function value. The point representing the center of mass (X_C) and can be calculated according to:

$$X_C = \frac{\sum_{i=1}^{N} \frac{1}{f^i} X_i}{\sum_{i=1}^{N} \frac{1}{f^i}} \tag{4}$$

where X_i is a point within an n-dimensional search space, f_i is a fitness function value of this point, and N is the population size. The other possibility is to select the fittest individual as the center of mass [4]. After calculating the center of mass, the algorithm generates new candidate solutions for the next Big Bang phase using a normal distribution around the previous center of mass according to:

$$X_i^{k+1} = X_C^k + r_j \alpha \frac{X_{\max} - X_{\min}}{k+1}, i = 1,..., N \tag{5}$$

where k is number of iteration, r_j is a random number from standard normal distribution which changes for every candidate, α is a parameter for limiting the size of search space and X_{max} and X_{min} are vectors consisting of maximal and minimal acceptable values for all variables. After the new population is generated, algorithm moves onto the next Big Crunch phase by calculating new center of mass. This sequence of explosion and contraction is repeatedly carried out until a stopping criterion has been met, whether it is reaching maximum number of iterations or obtaining a convergence.

Although the BB-BC method has been shown to outperform classical Genetic Algorithm for numerous benchmark functions [3] and obtains good results in the fine search around a local optimum (exploitation), it has some drawbacks in exploration (covering the whole search space). In order to overcome these drawbacks, we have implemented hybridization of BB-BC and some characteristics of Particle Swarm Optimization (PSO) proposed in [5] because this approach has been proven to be successful in solving different design problems [5, 6]. PSO is optimization algorithm inspired by birds flocking or fish schooling in search for food [7, 8], where every individual (particle) adjusts

its movements according to both its own experience and the population experience. This feature has been used for modifying the BB-BC into hybrid BB-BC (HBB-BC) by using not only the center of mass from the previous generation (X_c^k) but also the best position of a given individual solution up to iteration k ($X^{lbest(k)}$) and the best position in the whole population found up to the iteration k ($X^{gbest(k)}$) [5]:

$$X_i^{k+1} = \beta X_C^k + (1-\beta)\left[\gamma X^{gbest(k)} + (1-\gamma)X^{lbest(k)}\right] + r_j \alpha \frac{X_{\max} - X_{\min}}{k+1}, i = 1,...,N \tag{6}$$

where β and γ are adjustable coefficients for controlling the influence of the global and local best results $X^{gbest(k)}$ and $X^{lbest(k)}$, respectively.

In order to examine applicability of different BB-BC algorithms for optimum design of truss structures, we have analysed four variants of the algorithm. The first two algorithms, namely BB-BC1 and BB-BC2, are two variants of the classic BB-BC algorithm, where the latter one do not use center of mass calculated using Equation (4) but takes the fittest individual from the last iteration as the new center of mass. This was done in order to examine how this change would affect the calculation time and quality of solutions. The other two variants, namely HBB-BC1 and HBB-BC2, are hybrid algorithms where the Big Bang phase is conducted using Equation (6) instead of Equation (5). In the former one, the center of mass is calculated using Equation (4), while in the latter one the center of mass is the fittest individual from the previous iteration.

4 Comparison Criteria

Basically, every multi-objective optimizer aims at three goals: a) to find out the true Pareto front or to converge as close as possible to it; b) to cover as wide as possible span of solutions, and c) to discover solutions as diverse as possible along the obtained Pareto front. In order to provide a quantitative performance assessment for different multi-objective optimizing algorithms, it is necessary to establish exact criteria for measuring and comparing their effectiveness. A variety of such metrics and methods have been proposed in literature [9–11]. Method used in this study is based on a set of three metrics, namely: Coverage, Spacing and Maximum Spread [11, 12].

Coverage (C-metric) provides comparison between two Pareto fronts. If A and B be are two approximations to the Pareto front, then $C(A, B)$ is the percentage of the solutions in B that are dominated by at least one solution in A:

$$C_{(A,B)} = \frac{\left|\left\{u \in B \middle| \exists v \in A : v \ dom \ u\right\}\right|}{|B|} \tag{7}$$

Spacing (S-metric) indicates how evenly the solutions are distributed along the discovered Pareto-front:

$$S = \left[\frac{1}{n_{pf}} \sum_{i=1}^{n_{pf}} (d_i - \bar{d})^2\right]^{\frac{1}{2}}, \bar{d} = \frac{1}{n_{pf}} \sum_{i=1}^{n_{pf}} d_i \tag{8}$$

where n_{pf} is the number of members in Pareto front and d_i is the Euclidean distance (in the objective space) between the member i in Pareto front and its nearest member. A smaller value of S implies a more uniform distribution of solutions in Pareto front.

Maximum Spread (MS-metric) measures how "well" obtained Pareto front covers the true Pareto front:

$$MS = \left[\frac{1}{m} \sum_{i=1}^{m} \left[\frac{\min(f_i^{\max}, F_i^{\max}) - \max(f_i^{\min} - F_i^{\min})}{F_i^{\max} - F_i^{\min}} \right]^2 \right]^{\frac{1}{2}} \quad (9)$$

where m is the number of objectives, $f_{i\max}$ and $f_{i\min}$ are the maximum and minimum of the ith objective in obtained Pareto front, respectively, and $F_{i\max}$ and $F_{i\min}$ are the maximum and minimum of the ith objective in true Pareto front, respectively. A larger value of MS inicates a better spread of solutions. Since the true Pareto front in this study is not known, $F_{i\max}$ and $F_{i\min}$ are considered as the maximum and minimum of the ith objective in all obtained Pareto fronts by various algorithms.

5 Numerical example

Proposed algorithms were tested on a well-known example of 10-bars plane truss shown in Figure 1. The objectives are to minimize the weight of the structure and the displacement in y-direction at node 5 simultaneously using the cross-sectional areas of the 10 truss numbers as design variables. The cross-sectional area of every member is selected from a set of 32 discrete values predefined as {1.05, 1.16, 1.54, 1.69, 1.86, 1.99, 2.02, 2.18, 2.34, 2.48, 2.50, 2.70, 2.90, 3.10, 3.21, 3.30, 3.70, 4.66, 5.14, 7.42, 8.71, 8.97, 9.16, 10.00, 10.32, 12.13, 12.84, 14.19, 14.77, 17.10, 19.35, and 21.61}×10−3 m^2. External load is P = 444.5 kN, and material properties are modulus of elasticity E = 68.9 GPa and weight density ρ = 2712.6 kg⁄m3. Stress σ_j in each element has to be below the maximum allowable stress of σ_a = ±172 MPa.

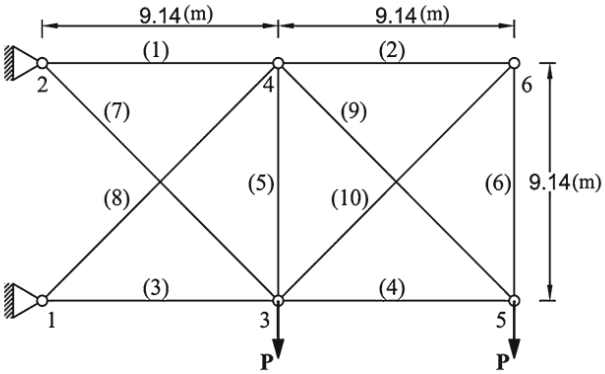

Fig.1. Geometry and loads for the 10-bars truss example.

6 Results and Discussion

In order to evaluate performance of described algorithms, they were run five times. Extreme solutions, required computational times for solving the problem example and number of solutions in each Pareto front are presented in Table 1 and the performance metrics values are presented in Table 2. Comparison of results for BB-BBC1 and BB-BC2, and for HBB-BC1 and HBB-BC2, respectively, indicates that using the fittest individual as the new center of mass instead of the one calculated using Equation (5) had no significant impact on the final solutions except the little improvement considering the calculation time.

On the other hand, comparison of results obtained by the algorithms BB-BC1 and HB-BBC1, and BB-BC2 and HB-BBC2, respectively, indicates that hybrid versions of the algorithm obtained better extreme results for the both fitness functions.

Table 1. Best results after five runs.

Algorithm	Best extreme solution 1 (kg, mm)	Best extreme solution 2 (kg, mm)
BB-BC1	(832.677; 135.98)	(5655.621, 29.91)
BB-BC2	(831.078; 140.85)	(5658.870, 29.89)
HBB-BC1	(825.251; 160.78)	(5698.841, 29.65)
HBB-BC2	(825.847; 159.24)	(5675.142, 29.68)

Differences in quality of obtained Pareto fronts can be further discussed according to the results of three considered performance metrics given in Table 4. Comparison of results obtained by BB-BC1 and BB-BC2, and HBB-BC1 and HBB-BC2, respectively, indicates that using the fittest individual as the new centre of mass had no significant impact on quality of results. However, hybridized versions of HB-BC outperformed the basic versions concerning all three metrics, so it can be concluded that hybridization is a good way of improving the performance of classic BB-BC algorithm.

Comparison of results obtained by BB-BC1 and BB-BC2, and HBB-BC1 and HBB-BC2, respectively, indicates that using the fittest individual as the new center of mass had no significant impact on quality of results. However, hybridized versions of HB-BC outperformed the basic versions concerning all three metrics, so it can be concluded that hybridization is a good way of improving the performance of classic BB-BC algorithm.

Table 2. Performance metrics values.

C-metric					
Algorithm	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
BB-BC1 vs. BB-BC2	0.23	0.24	0.17	0.22	0.27
BB-BC2 vs. BB-BC1	0.19	0.26	0.12	0.20	0.20
BB-BC1 vs. HBB-BC1	0.17	0.23	0.11	0.16	0.18
HBB-BC1 vs. BB-BC1	0.27	0.27	0.38	0.32	0.36
BB-BC1 vs. HBB-BC2	0.15	0.20	0.23	0.16	0.20
HBB-BC2 vs. BB-BC1	0.30	0.38	0.29	0.36	0.34
BB-BC2 vs. HBB-BC1	0.21	0.20	0.17	0.14	0.15
HBB-BC1 vs. BB-BC2	0.25	0.30	0.32	0.36	0.31
BB-BC2 vs. HBB-BC2	0.16	0.21	0.18	0.14	0.16
HBB-BC2 vs. BB-BC2	0.36	0.32	0.38	0.34	0.33
HBB-BC1 vs. HBB-BC2	0.21	0.25	0.22	0.26	0.24
HBB-BC2 vs. HBB-BC1	0.27	0.26	0.23	0.22	0.22
S-metric					
Algorithm	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
BB-BC1	28.042	29.982	28.674	27.054	29.002
BB-BC2	27.423	28.360	26.931	27.794	28.097
HBB-BC1	16.533	16.811	17.911	18.292	19.024
HBB-BC2	15.580	17.012	16.345	17.676	15.468
MS-metric					
Algorithm	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
BB-BC1	0.921	0.959	0.977	0.945	0.953
BB-BC2	0.9544	0.9638	0.970	0.9687	0.935
HBB-BC1	0.995	0.989	0.9848	0.991	0.990
HBB-BC2	1.000	0.998	0.995	0.989	1.0000

7 Conclusion

Appropriate choice of optimization techniques and tools is extremely important for overcoming insufficiencies and inadequacies of conventional trial-and-error approach in optimum structural design, especially for complex problems such as exploring numerous different possibilities of optimum truss design. In this paper a heuristic population-based search inspired by the Big Bang and Big Crunch theory (BB–BC) of the evolution of the universe is implemented for solving the classical truss optimization problem. Comparison of obtained numerical results for the benchmark problem indicates that proposed method is efficient and sufficiently reliable for solving complex problems in the field of optimum structural design and that hiridization of the multi-objective Big Bang-Big Crunch algorithm with Particle Swarm Optimizer outperforms basic versions of the algorithm both in calculation time and the quality of solutions.

References

1. F. Glover, M. Laguna, *Tabu Search* (Kulwer Academic Publishers, 1997)
2. T. Stutzle, *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications* (DISKI, Sankt Augustin, 1999)
3. O. K. Erol, I. Eksin, Advances in Engineering Software, **37**, 106–111 (2006)
4. H. M. Genc, I. Eksin, O. K. Erol, Turkish Journal of Electrical Engineering & Computer Sciences, **21**, 1359–1375 (2013)
5. A. Kaveh, S. Talatahari, Computers and Structures, **87(17–18)**, 1129–1140 (2009)
6. M. Sedighizadeh, M. Esmaili, Energy, **76**, 920–930 (2014)
7. J. Kennedy, R. Eberhart, Y. Shi, *Swarm intelligence* (Morgan Kaufmann Publishers, 2001)
8. J. Aghaei, K. M. Muttaqi, A. Azizivahed, M. Gitizadeh, Energy, **65**, 398–411 (2014)
9. K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms* (John Wiley & Sons, Chichester, 2001)
10. K. C. Tan, T. H. Lee, E. F. Khor, Artificial Intelligence Review, **17(4)**, 253–290 (2002)
11. E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. Fonseca, IEEE Transactions on Evolutionary Computation, **7(2)**, 117–132 (2003)
12. A. Kaveh, K. Laknejadi, Acta Mechanica, **224**, 343–364 (2013)