# Influence of adjusting Big Bang-Big Crunch Algorithm's parameters on resolving Constraint-based Optimization problems

**Hawar Bahzad Ahmad**

Nawroz University, College of Science, Computer Science Department, Duhok, Iraq

Received: 10.07.2024 Revised: 15.08.2024 Accepted: 09.09.2024

**ABSTRACT**
Metaheuristic is a collection of algorithms that are strongly discussed and used in the literature. Typically metaheuristics algorithm uses stochastic operators that makes each iteration unique, and often haves controlling parameters that have an impact on convergence process astheir effects mostly ignored in most literature related to optimization, which make it difficult to draw a conclusion. The Big Bang–Big Crunch (BB-BC) metaheuristic algorithm was introduced in this work to evaluate the performance of a metaheuristic algorithm in relation to its control parameter. And it shows the effects of variety in the values of BB-BC in solving A well-known engineering optimization problem known as“ tension/ compression spring design problem” that’s considered as a Single-Objective Constrained Optimization Problems.As part of the Experimental results, multiple beginning parameters for the BB-BC are tested. This is done in an attempt to discover the algorithm's optimum beginning parameters. And the minimum, maximum, and mean results of the penalized objectives functions are then computed.

**Keywords:** Metaheuristic Algorithms, Big Bang–Big Crunch, Tension/Compression Spring Design Problem, Constrained Optimization

## 1. INTRODUCTION

In the literature on discrete and combinatorial optimization, metaheuristic algorithms are widely used. They work by employing stochastic operators to explore the design space and direct the search toward optimum designs, emphasizing the requirement for statistical approaches to adequately evaluate their performance.Metaheuristic algorithms are often guided by a set of algorithmic parameters that may be altered to customize the search technique to the optimization task at hand. In this text, these parameters are referred to as control parameters.

A metaheuristic algorithm's performance is determined by three factors: 1) the optimization issue at hand, 2) the control parameter values, and 3) the random variability inherent in stochastic algorithms.

In the optimization field, in order to assess theperformance of newly proposed metaheuristic algorithms it is usually tested in to benchmark problem, in this study a well-known benchmark problem of the engineering field called Tension/Compression Spring Design Problem is used to evaluate the effect of variety in the initial value for Big Bang–Big Crunch (BB-BC) parameters.

## 2. LITERATURE REVIEW

In the past, methods with stochastic mechanisms were frequently referred to as "heuristic algorithms," but in more recent studies, they are referred to as "metaheuristics" that invented by Glover (1986)[1],as a combination of the words "meta" and "heuristic," where "meta" denotes a level above or beyond and "heuristic" generally refers to a "higher level of heuristics." Generally speaking, metaheuristic algorithms function as "master strategies that direct and modify other heuristics to yield answers beyond those that are typically obtained in a search for local optimality." These algorithms modify local search and randomization in a specific way. For challenging optimization issues, although there is generally no assurance of locating the best answers, but a good solution can be found in a reasonable amount of time[2].

The term "metaheuristic" refers to a higher-level technique or heuristic designed to search, find, generate, or pick a heuristic that could provide a viable solution to an optimization problem, particularly for large problems such as (NP-hard problem) or in the case of limited, incomplete, or imperfect information. a form of stochastic optimization used by metaheuristics makes the solution reliant on the collection of produced random variables[3]. Metaheuristics are generally more efficient in finding effective solutions

in combinatorial optimization than optimization algorithms, iterative techniques, or basic heuristics because they search across a much larger range of viable alternatives[4].

According to Almufti in 2019[5], there were more than 200 Metaheuristic algorithms have been developed to address a wide range of practical problems. Most algorithms get their inspiration from nature and incorporate elements of physical, biological, ethological, or swarm intelligence. Surprisingly, several of these methods, such as the genetic algorithm (GA) and particle swarm optimization (PSO)[6], are well recognized among specialists from other study domains in addition to computer scientists. In actuality, the widespread use of metaheuristic algorithms, particularly in engineering optimization problems, may be attributed to a number of factors, including their flexibility, gradient-free mechanism, and reliance on basic ideas for local optima avoidance.Because they are frequently based on plain or simple notions, the majority of nature-inspired metaheuristics algorithms are almost simple[7].

Metaheuristic algorithms are broadly grouped into differentcategories: evolutionary, swarm, physics or chemistry, and human behavior based algorithms. Evolutionary algorithms are just an adaptation to natural evolutionary processes[8]. The global optima are obtained in this type of algorithm by producing a new child that inherits the features and properties of parents by randomly selecting agents from the present population as parents and involving them in the production of offspring for the next generation[9]. Common evolutionary-based algorithms include evolutionary strategies, genetic algorithms, and genetic programming (GP)[6], ant colony optimization (ACO)[10]. Though they addressed various optimization issues, such as the infinite monkey theorem[11], Richard Dawkins' weasel[12], and the travelling salesman problem[13], the fundamental disadvantage of these methods is their computational cost.

Swarm based algorithms replicate the social and intellectual behaviour of a bunch of species (e.g., birds, insects, fish). For example, the famous particle swarm optimization technique was inspired by bird flight, while a new moth-flame optimization approach was inspired by moth navigation. Swarm-based algorithms tackle optimization problems by exhibiting self-organization, resilience, coordination, simplicity, and dispersal[14]. They also share information across several agents, are self-organized, co-evolve, and learn through iterations to execute efficient search operations. Furthermore, various agents may be parallelized, making large scale optimization more viable from an implementation standpoint[15]. Artificial bee colony optimization[16], ant colony optimization[13], whale optimization method[17], grasshopper optimization algorithm[18],Particle Swarm Optimization[14], Bat algorithm (BA)[19], Stochastic diffusion search[20], Cat Swarm Optimization (CSO)[21], artificial fish swarm algorithm[22], andelephant herding optimization[23]are some examples of swarm based algorithms.

Metaheuristic algorithms based on physics or chemistry are created differently, with inspiration drawn from known physics or chemistry occurrences. These algorithms often imitate physical or chemical laws such as electrical charges, river systems, chemical processes, gas pressure, gravity, and so on[9]. The gravitational search algorithm created by Rashedi et al. (2009)[24], models Newton's theory of gravitation, whereas the chemical reaction algorithm mimics chemical processes. Using control volume mass balance models, the equilibrium optimization method simulates the estimate process of equilibrium states. Magnetic charged system search, ions motion algorithm, atom search optimization, and henry gas solubility optimization are all physics/chemistry based metaheuristic algorithms.

Researchers have used a variety of strategies to handle constrained design optimization difficulties. Many optimization issues have been successfully solved using these approaches. Using the HS method, Lee and Geem (2004) tried to solve unconstrained, restricted, and structural optimization issues. Unified particle swarm optimization (UPSO) was used by Parsopoulos and Vrahatis to tackle four well-known constrained design optimization problems in 2005 tension/compression spring, welded beam, gear train, and pressure vessel. In 2022 Almufti solved welded beam by ABC[3].

Mezura-Montes and Coello (2005) used an evolutionary method to keep impractical solutions near to the practicable zone in order to address engineering design challenges without the need of a penalty function. Becerra and Coello (2006) proposed a cultural algorithm based on differential evolution to handle limited optimization issues (CDE).Liao (2010) assessed the efficacy of two hybrid differential evolution (DE) algorithms on 14 engineering design issues. The ABC method was used by Akay and Karaboga (2010) to test engineering design challenges[25]. Rao, Savsani, and Vakharia (2011) proposed and assessed the performance of the teaching-learning-based optimization (TLBO) method on design optimization issues. Zhang et al. (2013) tackled limited design optimization challenges by combining a tissue membrane system and DE in their differential evolution algorithm and tissue P systems (DETPS) method. Pavone, Narzisi, and Nicosia (2012) introduced an immunological approach for solving global numerical optimization problems for high-dimensional cases.Rios and Sahinidis (2013) used a derivative-free approach to solve bound-constrained optimization problems that needed just the availability of objective function values but no derivative knowledge. The authors presented an overview of derivative-free

algorithms before conducting a systematic analysis of 22 related implementations on a test set of 502 issues. On restricted design challenges, Rao and Waghmare (2014) evaluated the performance of an elitist teaching-learning-based optimization method (Elitist TLBO). The TLBO method is widely used by optimization experts[11].

In past years, tension/compression spring design problem has been solved by different researches using various algorithms, Coelho used QPSO algorithm [26], Omran et al. used CODEQ[27]Gandomi et al. used BAT[28] Saman M. Almufti used ABC [16],Saman M. AlmuftiusedVPS[7], He and Wangused CPSO[15], Modified FA usedMFA[29], Montes et al.used ES[30], Coello usedGA[31] Raj et al.used ADE[32].

## 3. Constrained Optimization Problem

Constrained optimization is a class of numerical techniques for problem-solving when the objective is to reduce total cost while relying on inputs with unfulfilled constraints or limits. The nature of the problem and the purpose for which it is to be solved might influence the approaches taken to solve it. These techniques have a broader connection to constraint fulfillment issues[7]. where the utilized must adhere to a set of requirements. Constrained optimization difficulties, however, it needsto reduce the total cost of the unmet limitations.The preferred method for identifying solutions that yield the issue's greatest possible performance under the given circumstances is thought to be metaheuristic algorithms[16].

An objective function or a collection of objective functions can be used to characterize how well the task performs. Therefore, there are two categories of constrained optimization problems:

- Single-Objectives Constrained Optimizations Problems (SOCOP).
- Multi-Objectives Constrained Optimizations Problems (MOCOP).

Where, both maximal and minimal optimization of these optimization problem are possible.

In this paper,aSingle-Objective Constrained Optimization Problems known as tension/compression spring design problem is used to evaluate the effect of variety in the initial parameter of BB-BC.

### 3.1 tension/compression spring design problem

Vibration is a mechanical phenomenon in which oscillations occur around an equilibrium situation. It is a well-known problem that falls under the engineering category of Single-Objective Constrained Optimization Problems (SOCOP). Vibration can have two forms: Natural and forced vibration are the two forms of vibration[7].

A continuous constrained problem is shown by the tension/compression spring design problem (TCSD). The goal is to reduce the volume of the coil spring while under continual tension/compression stress. This problem is caused by three design considerations. They are as follows:

- $P = x1 \in [2, 15]$
- $D = x2 \in [0.25, 1.3]$
- $d = x3 \in [0.05, 2]$.

The active coil count of the spring is denoted by P, the winding diameter by D, and the wire diameter by d. These parameters are used to minimize weight while adhering to limits on flow frequency, shearing stress, and minimal deflections see Fig.1.

The cost function Eq(1), which must be minimized utilizing restrictions g1, g2, g3, and g4 in Eq(2), may be utilized to formally define the TCSD issue as follows Eq(2):

$$f(x) = (X_3 + 2)X_2X_1^2 \tag{1}$$

The values of $X_1, X_2,$ and $X_3$limit the design's available space. The Lower-Boundary (Lb) = [0.05,0.25,2] and the Upper-Boundary (Ub) = [2,1.3,15] limit the range of design variables. utilizing the descriptions for the COP and penalty obtained in Eq(1). Shear stress, surge frequency, and minimum deflection all have four restrictions. Eq(2) shows the four constraints, three design variables, and one objective function.

$$\begin{aligned} g_1(x) &= 1 - \frac{X_2^3X_3}{72785X_1^2} \quad \leq 0 \\ g_2(x) &= \frac{4X_2^2 - X_1X_2}{12566(X_2X_1^3) - X_1^4} + \frac{1}{5108X_1^2} - 1 \leq 0 \\ g_3(x) &= 1 - \frac{140.45X_1}{X_2^2X_3} \quad \leq 0 \\ g_4(x) &= \frac{X_1 + X_2}{1.5} - 1 \quad \leq 0 \end{aligned} \tag{2}$$

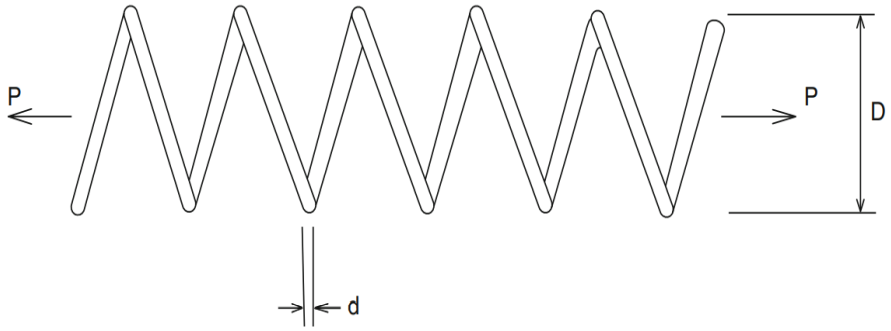



**Figure 1:** Schematic of the tension/compression spring

in an effort to evaluate the performance of the BB-BC metaheuristic algorithm. The design issue for tension/compression springs is chosen. Which is an established engineering optimization issue.

## 4. Big Bang–Big Crunch (BB-BC) algorithm

Erol and Eksin [1] have developed a fresh and simple nature-based or physics-based metaheuristic that was inspired by the energy dissipation in the form of change from an ordered state to a disordered or chaotic one called The Big Bang-Big Crunch (BB-BC) algorithm, which based on evolutionary theory[33].
The Big Bang Theory explains the universe's genesis. According to this idea, during the Big Bang phase, particles lose energy and are driven toward irregularity, but during the Big Crunch phase, they converge in a certain direction. Similar to previous population-based metaheuristics, BB-BC begins with a random collection of initial candidate solutions, referred to as the Big Bang. In actuality, each Big Bang phase is preceded by a Big Crunch phase, with the exception of the initial population, which must be produced randomly within the search space. After each Big Bang phase, there should be a Big Crunch phase to identify a convergence operator by which particles will be ordered in the upcoming Big Bang phase[34]. The convergence operator might be the weighted average of the candidate solution locations or the position of the candidate solution with the highest score. In the cyclic body of the algorithm, these two contraction (Big Crunch) and dispersal (Big Bang) phases are performed in succession to fulfill a stopping criterion and direct the particles toward the global optimum.
There is a similarity between the Big Bang Theory and a population-based metaheuristic whether seen from a physical, natural, or astronomical perspective. Each particle represents a member of the algorithm's population or a proposed solution. In the Big Bang phase, a given number of particles are repeatedly updated in the search space with step sizes depending on the convergence operators of the Big Crunch phase in an effort to converge on the global optimum of the problem.
As with previous population-based metaheuristics, the Big Bang-Big Crunch (BB-BC) method begins with a collection of randomly generated beginning solutions in the search space. Each cycle of the algorithm consists of two phases: first, the Big Crunch phase, in which a converging operator is created, and second, the Big Bang phase, in which particles in the search space are updated with step sizes in the region of the converging operator created in the first phase. Consider a specific number of particles (nP) as the population or candidate solutions matrix (P), its associated penalized objective function (PFit), and the best observed particle in each iteration (bestP) with the least penalized objective function value[35].
The convergence operator based on the Big Crunch phase may be described as the weighted average of candidate solution locations, often known as the center of mass (CM), or the position of the best candidate solution (bestP). For minimization issues, CM is expressed as follows[29]:

$$CM(i) = \sum_{j=1}^{nP} \left( \frac{P(j,i)}{PFit(j)} \right) / \sum_{j=1}^{nP} \left( \frac{1}{PFit(j)} \right), \qquad i = 1, \dots, nv \tag{3}$$

The Big Bang phase may now be initiated. In the original BB-BC [1], particles are simply updated with respect to the previously established center of mass (CM) or position of the best particle (bestP) by displacing by a random fraction of the permitted step size indicated by the upper (Ub) and lower (Lb) limit of design variables:

$$newP = (CM \text{ or } bestP) + \frac{rand * (Ub - Lb)}{nIT} \tag{4}$$

where rand is a uniformly distributed random number (0, 1). The step size is also divided by the number of algorithm iterations or number of Big Bang phases (NITs) in order to establish the effective search range around the global optimum or center of mass in order to restrict the search as the algorithm progresses. Clearly, the method contains two parameters that are necessary for all metaheuristics: the population size and the maximum number of algorithm iterations as a stopping criterion. Camp [2] provided a novel formulation with two extra parameters for the Big Bang phase and demonstrated its effectiveness. The revised formulation is as follows[35]:

$$\text{newP(i)} = (\beta * \text{CM} + (1-\beta) * \text{bestP} + \frac{\text{rand} * \alpha * (\text{Ub} - \text{Lb})}{\text{nIT}}, \quad \text{i} = 1, \ldots, \text{nP} \tag{5}$$

This updated formulation is used and encoded inside this section. Notably, the BB-BC algorithm does not need a replacement approach. In other words, particles depart their place regardless of whether their present position is advantageous.
The pseudo code of the method is supplied below, and the BB-BC flowchart is seen in Fig2.

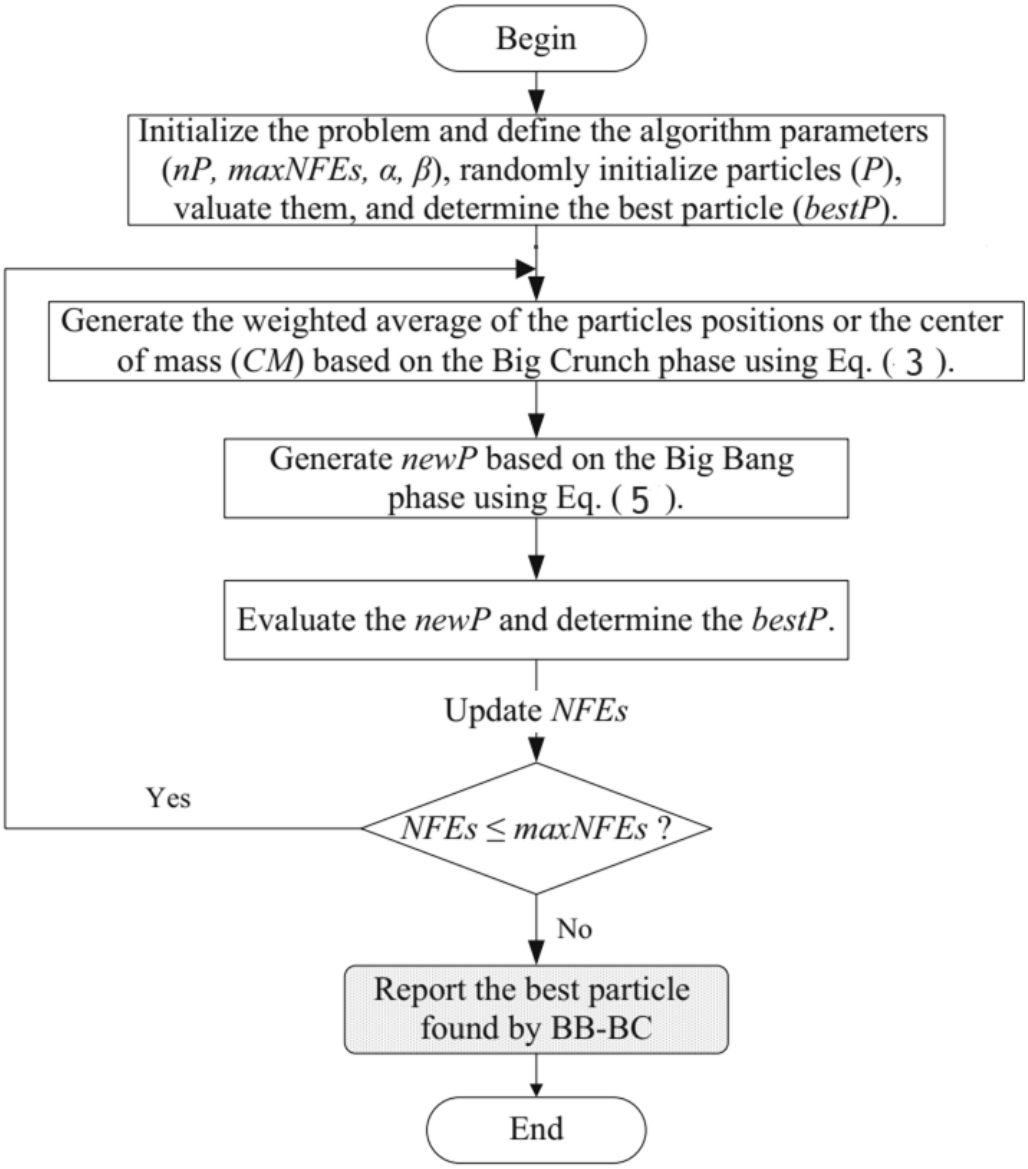



**Figure 2:** BB-BC Algorithm Flowchart

## 5. Experimental results

In this section, the results of using the Big Bang–Big Crunch (BB-BC) algorithm to solve the tension/compression spring design problem (TCSD) with different starting values for the algorithm's parameters are presented. The TCSD refers to the problem of designing springs that can be compressed

or stretched.
In most cases, BB-BC is dependent on a number of constant variables that have a direct impact on the way in which the results converge to the best possible value. In this paper, various values for the BB-BC variables are tested to demonstrate the effects of those parameters on BB-BC performance. The paper also investigates the exploration and exploitation capabilities of the algorithm in order to achieve BB-BC's highest level of effectiveness in resolving the (TCSD).

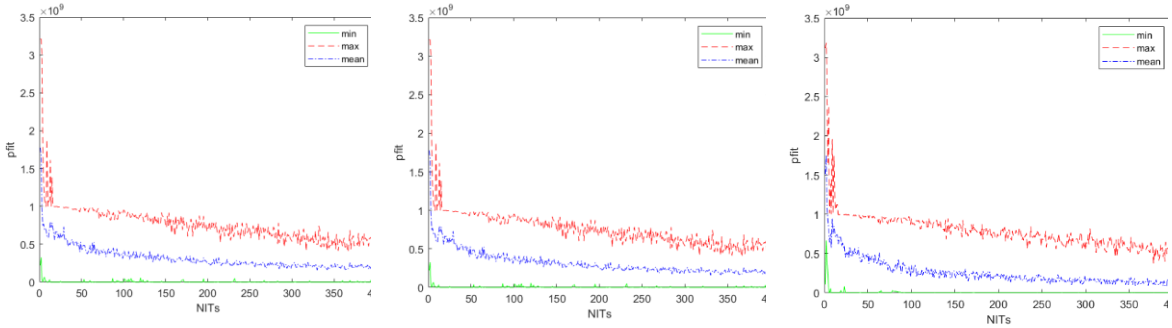
### A. BB-BCAlpha parameter

Alpha Parameter is used by the BB-BC algorithm to restrict the initial search space and create random first solutions. Table 1 illustrates the influence of various Alpha values on the BB-BC algorithm's ability to solve the TCSD problem. It depicts the convergence history for a single run of the algorithm with the same initial population and varying Alpha values (0.5, 1, 1.5). In these runs, the value partners Beta=0.2 and nP=50 should be stated.

**Table 1:** BB-BC results with Beta=0.2nP=50, maxNFEs=20000 and various value of Alpha

| Alpha | X1 | X2 | X3 | F(X) | time |
|---|---|---|---|---|---|
| 0.5 | 0.0547585388455896 | 0.429385226928658 | 8.38852620024366 | 0.012851 | 0.4103 |
| 1 | 0.0563538157370893 | 0.473178333816682 | 6.94608047432395 | 0.013443 | 0.4242 |
| 1.5 | 0.0546889207450174 | 0.437465828002190 | 7.93161352072325 | 0.012978 | 0.4688 |

Table 1 ,shows the minimum fitness of solving TCSD with three constraint values (X1, X2, and X3). It shows that when Alpha=0.5, the result of the BB-BC algorithm is optimal.




**(a) Alpha=0.5** **(b) Alpha=1** **(c) Alpha=1.5**

**Figure 3:** effect of Alpha parameter on BB-BC Algorithm results

Figure 3 illustrates how different Alpha values may have a significant impact on the outcomes of BB-BC algorithms. It is important to keep in mind that the magnitude of these impacts grows proportionately with both F(X) and the amount of elapsed time of the algorithm.

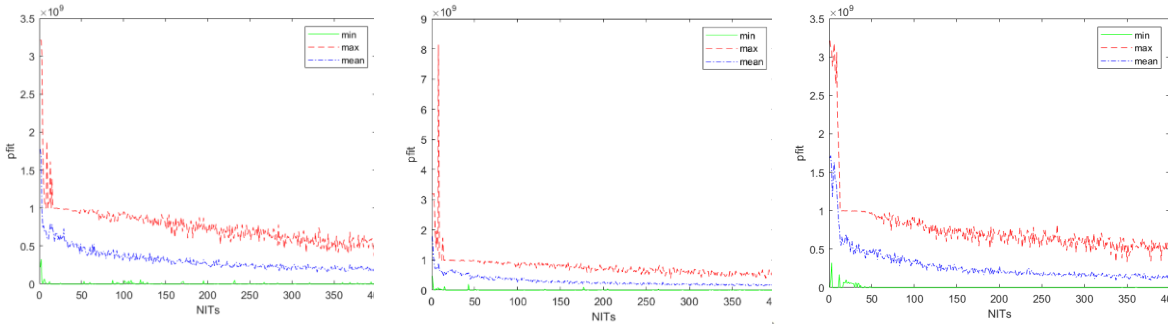
### B. BB-BCBeta parameter

beta Parameter uses for regulating the weighted average of the particle locations or the center of mass (CM) and the best particle. Table 2 illustrates the influence of various Beta values on the BB-BC algorithm's ability to solve the TCSD problem. It depicts the convergence history for a single run of the algorithm with the same initial population and varying Beta values (0.2, 0.5, and 0.8). In these runs, the value partners Alpha=0.5 and nP=50 should be stated.

**Table 2:** BB-BC results with Alpha=0.5nP=50, maxNFEs=20000 and various value of Beta

| Beta | X1 | X2 | X3 | F(X) | time |
|---|---|---|---|---|---|
| 0.2 | 0.0547585388455896 | 0.429385226928658 | 8.38852620024366 | 0.012851 | 0.4103 |
| 0.5 | 0.0525869000637171 | 0.377457944586457 | 10.4058927770286 | 0.012766 | 0.4383 |
| 0.8 | 0.0517586917866834 | 0.357749310168480 | 12.2674695965016 | 0.013026 | 0.4613 |

Table 2 ,shows the minimum fitness of solving TCSD with three constraint values (X1, X2, and X3). It shows that when Beta=0.5, the result of the BB-BC algorithm is optimal.




**(a) Beta=0.2** **(b) Beta=0.5** **(c) Beta=0.8**

**Figure 4:**effect of Beta parameter on BB-BC Algorithm results

Figure 4 illustrates how different Beta values may have a significant impact on the outcomes of BB-BC algorithms. It is important to keep in mind that the magnitude of these impacts of F(X) us optimal when Beta= 0.5.

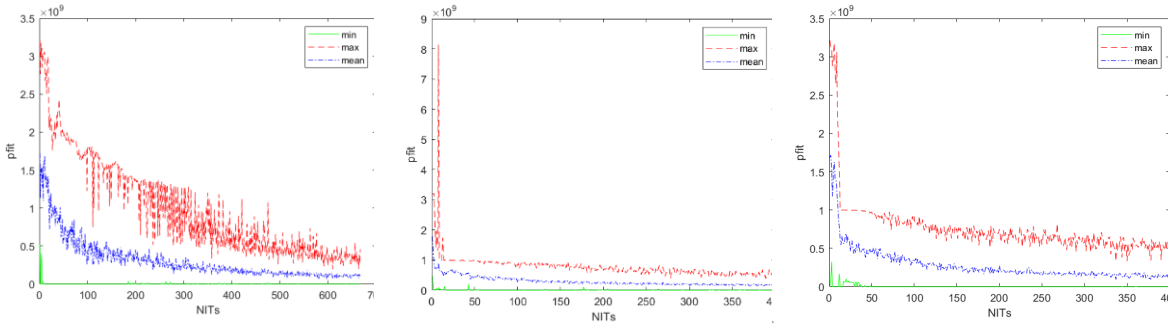
### C. BB-BCnP parameter

nP represents the number of Particles that participate in the search process. Table 3 illustrates the influence of various Beta values on the BB-BC algorithm's ability to solve the TCSD problem. It depicts the convergence history for a single run of the algorithm with the same initial population and varying nP values (30, 50, and 100). In these runs, the value partners Alpha=0.5 and Beta=0.5 should be stated.

**Table 3:** BB-BC results with Alpha=0.5Beta=0.5, maxNFEs=20000 and various value of nP

| **nP** | X1 | X2 | X3 | **F(X)** | **time** |
|---|---|---|---|---|---|
| 30 | 0.0637476877213870 | 0.725844574599826 | 3.18909476301756 | 0.015285 | 0.3923 |
| 50 | 0.0525869000637171 | 0.377457944586457 | 10.4058927770286 | 0.012766 | 0.4383 |
| 100 | 0.0500000000000000 | 0.316672814636553 | 14.1413517399200 | 0.012748 | 1.9538 |

Table 3 ,shows the minimum fitness of solving TCSD with three constraint values (X1, X2, and X3). It shows that when nP=100, the result of the BB-BC algorithm is optimal.




**(a) nP=30** **(b) nP=50** **(c) nP=100**

**Figure 5:**effect of Beta parameter on BB-BC Algorithm results

Figure 5 illustrates how different nP values may have a significant impact on the outcomes of BB-BC algorithms. It is important to keep in mind that the magnitude of these impacts of F(X) us optimal when nP= 100.

### 4.3. BB-BCresult with optimal parameter value

According to the findings from the preceding section tables (1, 2, and 3)shows that when the BB-BC parameters Alpha, Beta, and nP are set to 0.5, 0.5, and 100, respectively, and the Maximum number of Objective Function Evaluations (maxNFE) is set to 20,000, the BB-BC algorithm converges and produces a near-optimal solution for the tension/compression spring design issue.
Figure (6), displays the min, max and mean convergences history

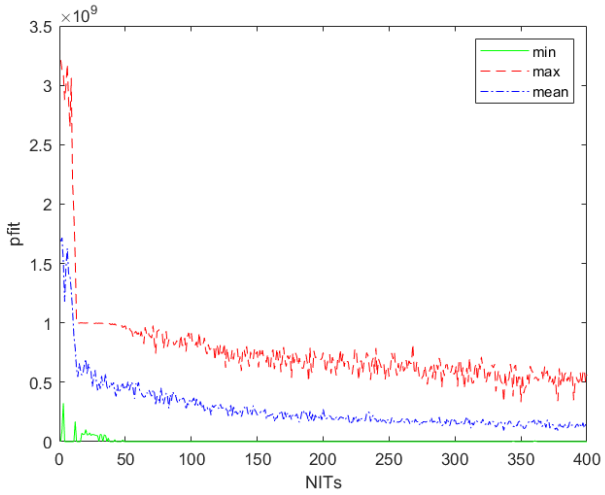



**Figure 6:** convergence history of TCSD solved by BB-BC min, max, and mean

Table 4, demonstrates the minimum, maximum, and mean outcomes of using the BB-BC method to solve TCSD.

**Table 4:** Proposed BB-BC algorithm results

| ABC | minPFit) | maxPFit | meanPFit | Time Elapsed |
|---|---|---|---|---|
| maxNEFs=200000<br>nP=100<br>Alpha=0.5<br>Beta=0.5 | 0.012748 | 807863620 | 192428011 | 1.9538 |

Table 2, compares the proposed algorithm results with the previous study results that are founded by different metaheuristics algorithms. it show the values of the three TCSD constraints (X1, X2, X3) and the number of fitness results.

## 6. CONCLUSION

Metaheuristic algorithms are commonly used to tackle issues with constraints. They were motivated by a natural occurrence.

In this paper, the Big Bang–Big Crunch (BB-BC) metaheuristic algorithm is used to solve the tension/compression spring design problem, a single objective restricted optimization problem in engineering.As described in preceding sections, the BB-BC algorithm relies on a number of factors to converge to the optimal solution. The purpose of this study is to identify the ideal value for the Alpha, nP, and Beta characteristics by evaluating several values.

- For the Alpha attribute, where MaxNFEs = 20000, nP = 50, and Beta= 0.2 are held constant, the convergence histories for a single trial run from the same beginning population for different values of the Alpha parameter range between 0 and 1 for various Alpha parameter values (0.5, 1, and 1.5). If Alpha is set to 0.5, the procedure operates more efficiently.
- For the Beta attribute, where MaxNFEs = 20000, nP = 50, and Alpha= 0.5 are held constant, the convergence histories for a single trial run from the same beginning population for different values of the Beta parameter range from 0.0 to 1.0. (0.2, 0.5, and 0.8). With a Beta value of 0.5, the approach operates more efficiently.
- For the nP attribute, by fixing MaxNFEs = 20000, Alpha = 0.5, and beta = 0.5, the convergence histories for a single trial run from the same beginning population for different values of the nP parameter range from 0 to 1 for MaxNFEs = 20000, Alpha = 0.5, and beta = 0.5. (30 50, and 100). When nP is set to 100, the technique runs more efficiently.

Finally, all variables are restored to their optimal levels, and the optimal (minimum, maximum, and average) PFit convergent value is determined. And a comparison of the results of 10 metaheuristic algorithms for tension/compression spring design is shown.

**REFERENCES**


[1] F. Glover, "Future paths for integer programming and links to artificial intelligence.," Computers & Operations Research, vol. 13, no. 5, p. 533–549, 1986.

[2] V. A. Saeed, S. M. Almufti and R. B. Marqas, "Taxonomy of bio-inspired optimization algorithms," Journal of Advanced Computer Science & Technology, vol. 2, no. 23-31, p. 8, 2019.

[3] S. M. Almufti, "U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem".

[4] R. R. Asaad and N. L. Abdulnabi, "Using local searches algorithms with Ant colony optimization for the solution of TSP problems," Academic Journal of Nawroz University, vol. 7, no. 3, pp. 1-6, 2018.

[5] S. M. Almufti, "Historical survey on metaheuristics algorithms," International Journal of Scientific World, vol. 7, no. 1, pp. 1-12, 2019.

[6] A. Y. Zebari, H. K. Omer and S. M. Almufti, "A comparative study of particle swarm optimization and genetic algorithm," Journal of Advanced Computer Science & Technology, vol. 8, no. 2, pp. 40-45, 2019.

[7] S. M. Almufti, "Vibrating Particles System Algorithm performance in solving Constrained Optimization Problem," Academic Journal of Nawroz University, vol. 11, no. 3, pp. 231-242, 2022.

[8] E. H. Houssein, M. R. S. K. Hussain, W. Zhu, H. Shaban and d. M. Hassaballah, "Optimal Sink Node Placement in Large Scale Wireless Sensor Networks Based on Harris' Hawk Optimization Algorithm," IEEE Access, vol. 8, pp. 19381-19397, 2020.

[9] E. H. Houssein, M. R. Saad, F. A. Hashim and H. Shaban, "Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems," Engineering Applications of Artificial Intelligence, vol. 94, 2020.

[10] S. M. Almufti, R. B. Marqas, P. S. Othman and A. B. Sallow, "Single-based and Population-based Metaheuristics for Solving NP-hard Problems," Iraqi J Sci, vol. 62, no. 5, pp. 1-11, 2021.

[11] C. R. S. Banerji, T. Mansour and S. Severini, "A notion of graph likelihood and an infinite monkey theorem," Journal of Physics A: Mathematical and Theoretical, 2014.

[12] M. Shermer, "To be or not to be a weasel: Hamlet, intelligent design, and how evolution works.," Skeptic (Altadena, CA), vol. 9, no. 4, 2002.

[13] S. M. Almufti, "Using Swarm Intelligence for solving NP-Hard Problems," Academic Journal of Nawroz University, vol. 6, no. 3, pp. 46-50, 2017.

[14] D. A. Zebari, D. Q. Zeebaree, J. N. Saeed, N. A. Zebari and A.-Z. Adel, "Image steganography based on swarm intelligence algorithms: A survey," Test engineering & managment, vol. 7, no. 8, 2020.

[15] Q. W. L. He, "An Effective Co-Evolutionary Particle Swarm Optimization for Constrained Engineering Design Problems," Engineering Applications of Artificial Intelligence, vol. 20, pp. 89-99, 2007.

[16] S. M. Almufti, A. A. Alkurdi and E. A. Khoursheed, "Artificial Bee Colony Algorithm Performances in Solving Constraint-Based Optimization Problem," Telematique, vol. 21, no. 1, p. 6785–6799, 2022.

[17] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," Advances in Engineering Software, vol. 95, pp. 51-67, 2016.

[18] Y. Meraih, A. B. G. S. Mirjalili and A. Ramdane-Cherif, "Grasshopper Optimization Algorithm: Theory, Variants, and Applications," IEEE Access, vol. 9, pp. 50001-50024, 2021.

[19] A. Y. Zebari, S. M. Almufti and C. M. Abdulrahman, "Bat algorithm (BA): review, applications and modifications," International Journal of Scientific World, vol. 8, no. 1, pp. 1-7, 2020.

[20] al-Rifaie, M. Majid and B. J. Mark, "Stochastic Diffusion Search Review," Journal of Behavioral Robotics, vol. 4, no. 3, pp. 155-173, 2013.

[21] R. R. Ihsan, S. M. Almufti, B. M. Ormani, R. R. Asaad and R. B. Marqas, "A Survey on Cat Swarm Optimization Algorithm," Asian Journal of Research in Computer Science, vol. 10, no. 2, pp. 22-32, 2021.

[22] C. Zhang, F. -m. Zhang, F. Li and H. s. Wu, "Improved artificial fish swarm algorithm," in 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, China, 2019.

[23] B. W. Salim, S. M. Almufti and R. R. Asaad, "Review on elephant herding optimization algorithm performance in solving optimization problems," International Journal of Engineering & Technology, vol. 7, no. 8, pp. 6109-6114, 2019.

[24] H. N.-p. S. S. Esmat Rashedi, "GSA: A Gravitational Search Algorithm," Information Sciences, vol. 179, no. 13, pp. 2232-2248, 2009.

[25] A. Kattan and R. A. Alrawi, "Pressure Vessel Design Using the Dynamic Self-adaptive Harmony Search Algorithm," in the Fourth International Conference on Digital Information Processing and Communications (ICDIPC2014), Kuala Lumpur, Malaysia, 2014.

[26] L. S. Coelho, "Gaussian Quantum-Behaved Particle Swarm Optimization Approaches for Constrained Engineering Design Problems," Expert Systems with Applications, vol. 37, pp. 1676-1683, 2010.

[27] M. G. H. S. A. Omran, "Constrained optimization using CODEQ, Chaos," Solitons & Fractals, vol. 42, pp. 662-668, 2009.

[28] A. H. Y. X. S. A. A. H. T. S. Gandomi, "Bat Algorithm for Constrained Optimization Tasks," Neural Computing and Applications, vol. 22, no. 6, pp. 1239-1255, 2013.

[29] M. J. Kazemzadeh-Parsi, "A Modified Firefly Algorithm for Engineering Design Optimization Problems," IJST, Transactions of Mechanical Engineering, vol. 38, pp. 403-421, 2014.

[30] E. M. C. C. A. C. Montes, "An Empirical Study About The Usefulness of Evolution Strategies to Solve Constrained Optimization Problems," International Journal of General Systems, vol. 37, pp. 443-473, 2008.

[31] A. C. C. Carlos, "Use of a self-adaptive penalty approach for engineering optimization problems," Computers in Industry, vol. 41, p. 113–127, 2000.

[32] K. H. S. R. S. M. G. S. D. A. P. C. Raj, "An Evolutionary Computational Technique for Constrained Optimisation in Engineering Design," Journal of the Institution of Engineers India Part Me Mechanical Engineering Division, vol. 856, pp. 121-128, 2005.

[33] Y. K. Qawqzeh, G. Jaradat, A. Al-Yousef, A. Abu-Hamdah, I. Almarashdeh, M. Alsmadi, M. Tayfour, K. Shaker and F. Haddad, "Applying the big bang-big crunch metaheuristic to large-sized," International Journal of Electrical and Computer Engineering (IJECE), vol. 10, no. 3, p. 2484~2502, 2020.

[34] G. M. Jaradat and M. Ayob, "Big Bang-Big Crunch Optimization Algorithm to Solve the Course Timetabling Problem," IEEE, pp. 1448-1452, 2010.

[35] A. Kaveh and T. Bakhshpoori, Metaheuristics: Outlines, MATLAB Codes and Examples, springer, 2019.

[36] S. L. Andresen, "John McCarthy: Father of AI," IEEE INTELLIGENT SYSTEMS, pp. 84-85, 2002.

[37] J. W. a. G. Beni, "Cellular Robot System with Stationary Robots and its application to Manufacturing Lattices," in 4th IEEE International Symposium on Intelligent Control, Albany, 1989.

[38] D. Karaboga, "AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION," Erciyes University, Engineering Faculty, Computer Engineering Department, Erciyes , 2005.

[39] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," Applied Mathematics and Computation, vol. 214, no. 1, p. 108–132, 2009.