Chapter 18
Big Bang–Big Crunch Algorithm

Abstract In this chapter, the big bang–big crunch (BB–BC), a global optimization method inspired from one of the cosmological theories known as closed universe, is introduced. We first, in Sect. 18.1, describe the background knowledge regarding the big bang and big crunch. Then, Sect. 18.2 details the fundamentals of BB–BC, the selected variants of BB–BC, and the representative BB–BC application, respectively. Finally, Sect. 18.3 draws the conclusions of this chapter.

18.1 Introduction

Cosmological theory is an exciting subject, because it shows how the universe happens, moves, and revolutions. One of the fascinating topics is where all of the stars and galaxies came from (and how, and why)? This question has long been explored by the physics. For example, two famous physics, i.e., Sir Isaac Newton and Albert Einstein, believed that the universe is unchanging and introduced a term, called cosmological constant. However, this would prove to be a mistake. In 1929, astronomer Edwin Hubble discovered that the universe was expanding. He found that in the early universe, gravity was very strong, as a result of the concentration of matter in a very small space—so small, in fact, that it was compressed down to a single point. Thus, it would suffer an incredible pressure and has expanded ever since, known as the big bang. This event was controversial until 1965, when an accidental discovery supported the theory. Today, the most advanced astronomical observations show that the big bang theory is likely true.

Scientists were originally very upset by the big bang theory, because they believed in an eternal universe (i.e., the universe does not change over time). However, they concerned soon with another question of what is the ultimate fate of the universe? One idea that was popular was that the universe would expand until gravity began to pull it back, resulting in a big crunch, where all matter returned to a single unified point—and then the cycle of expansion would start all over again. This hypothesis is known as closed universe. What happens after that? We cannot exactly tell for now.

B. Xing and W.-J. Gao, *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms*, Intelligent Systems Reference Library 62,
DOI: 10.1007/978-3-319-03404-1_18, © Springer International Publishing Switzerland 2014

18.1.1 Big Bang

Literally, every bit of matter and energy in our universe was created through a singular event (i.e., the unimaginable crucible of heat and light) that we call the big bang (Bauer and Westfall 2011). Just for the record, it was neither big (in fact, it was very small and fit onto the head of a pin), nor was there a bang. This event happened $(13.73 \pm 0.12) \times 10^9$ years ago. Although this theory is not perfect, and over time physics made efforts in order to make it more consistent. One thing is true that the universe was well on its way to becoming what we observe today: filled with galaxies, stars, planets, and all other sorts of strange and exotic things.

18.1.2 Big Crunch

One hypothesis that the future of the universe is called big crunch model, which means that the universe contracts back into a point of mass. This will proceed almost exactly like the big bang, except in reverse. Whether the expansion of the universe will take place forever or will stop some day, it depends on the quantity of matter it has (Scalzi 2008).

18.2 Big Bang–Big Crunch Algorithm

18.2.1 Fundamentals of the Big Bang–Big Crunch Algorithm

Inspired from one of the cosmological theories that universe was "born" in the big bang and might "die" in the big crunch, Erol and Eksin (2006) proposed a new algorithm, namely, the big bang–big crunch (BB–BC) algorithm. In general, the proposed algorithm includes two successive phases. In the big bang phase (corresponding to the disorder caused by the energy dissipation in nature), random points are generated; whereas, in the big crunch phase (corresponding to the order due to gravitational attraction), those points shrank to a single representative point via a centre of mass or minimal cost approach.

18.2.1.1 Big Bang Phase

Just like the expansion of the universe, the main purpose of the big bang phase in BB–BC is to create initial populations. The initial position of each input is generated randomly over the entire search space. Once the population pool is created, fitness values of the individuals are calculated (Genç et al. 2010).

18.2.1.2 Big Crunch Phase

If enough mass and energy (i.e., inputs) is in the universe (i.e., search space), then that mass and energy may cause enough attraction to halt the expansion of the universe and reverse it—bringing the entire universe back to a single point. Similarly, in the big crunch phase, a contraction procedure is applied to form a centre or a representative point for further big bang operations. In other words, the big crunch phase is a convergence operator that has many inputs but only one output, which called "centre of mass". Here, the term mass refers to the inverse of the fitness function value (f^i). The point representing the centre of mass that is denoted by x^c and calculated according to Eq. 18.1 (Erol and Eksin 2006):

$$\vec{x}^c = \frac{\sum_{i=1}^{N} \frac{1}{f^i} \vec{x}^i}{\sum_{i=1}^{N} \frac{1}{f^i}}, \tag{18.1}$$

where x^i is a point within an n-dimensional search space generated, f^i is fitness function value of this point (such as cost function), N is the population size in big bang phase.

Instead of the centre of mass, the best fit individual (i.e., the lowest f^i value) can also be chosen as the starting point in the big bang phase.

The new generation for the next iteration in the big bang phase is normally distributed around the centre of mass, using Eq. 18.2 (Erol and Eksin 2006):

$$x^{new} = x^c + \frac{lr}{k}, \tag{18.2}$$

where x^c stands for centre of mass, l is the upper limit of the parameter, r [or $N(0, 1)$] is a normal random number generated according to a standard normal distribution with mean (μ) zero and standard deviation (σ) equal to one, and k is the iteration step. Then new point (x^{new}) is upper and lower bounded.

Summarizing the steps in the standard BB–BC algorithm yields to Erol and Eksin (2006):

- Step 1: Initiation population of N candidate solution is randomly generated all over the search space.
- Step 2: The fitness function value (f^i) corresponding to each candidate solution is calculated.
- Step 3: The N candidate solutions are contracted into the centre of mass (x^c), either by using the Eq. (18.1) or by choosing the point that has lowest value after the calculation in Step 2.
- Step 4: New population of solutions is generated around x^c by adding or subtracting a random number whose value decreases by increasing the iterations elapsed.
- Step 5: Check if maximum iteration is reached; go to Step 2 for new beginning. If a specified termination criteria is satisfied stop and return the best solution.

18.2.2 Performance of BB–BC

To evaluate the performance of the BB–BC algorithm, (Erol and Eksin 2006) proposed six test functions, namely, Sphere function, Rosenbrock function, Step function, Ellipsoid function, Rastrigin function, and Ackley function. Compared with combat genetic algorithm (CGA), the BB–BC algorithm presented a better results of finding the global best solution.

18.2.3 Selected BB–BC Variants

Although BB–BC algorithm is a new member of computational intelligence (CI) family, a number of BB–BC variations have been proposed in the literature for the purpose of further improving the performance of BB–BC. This section gives an overview to some of these BB–BC variants which have been demonstrated to be very efficient and robust.

18.2.3.1 Hybrid BB–BC Algorithm

In Kaveh and Talatahari (2009, 2010b), the authors developed one of the first BB–BC hybrids, called hybrid BB–BC (HBB–BC). Overall, the HBB–BC introduced two improvements: using the particle swarm optimization (PSO) capacities to improve the exploration ability of BB–BC algorithm, and using sub-optimization mechanism (SOM) to update the search-space of BB–BC algorithm. Compared with the standard BB–BC and other conventional CI optimization methods such as genetic algorithm (GA), ant colony optimization (ACO), PSO, and harmony search (HS), the HBB–BC performed better.

In general, there are also two phases involved in HBB–BC: a big bang phase where candidate solutions are randomly distributed over the search space, and a big crunch phase working as a convergence operator where the centre of mass is generated. Compared with standard BB–BC algorithm, the main difference is that the HBB–BC employed the PSO capacities to improve the exploration ability. Kaveh and Talatahari (2009) pointed out that the reason to select PSO as the first reformation due to at each iteration, the particle moves towards both local best (i.e., a direction computed from the best visited position), and global best (i.e., the best visited position of all particles in its neighbourhood). Inspired by that, the HBB–BC approach not only uses the centre of mass but also utilizes the best position of each candidate and the best global position to generate a new solution. The calculation formulas in big crunch phase are as follows:

- The centre of mass can be computed via Eq. 18.3 (Kaveh and Talatahari 2009):

$$x_i^{c(k)} = \frac{\sum_{j=1}^{N} \frac{1}{f^i} x_i^{(k,j)}}{\sum_{j=1}^{N} \frac{1}{f^i}}, \quad i = 1, 2, \ldots, ng, \tag{18.3}$$

where $x_i^{c(k)}$ is the ith component of the jth solution generated in the kth iteration; f^i is fitness function value of this point (such as cost function); N is the population size in big bang phase.

- The new generation for the next iteration in the big bang phase is normally distributed around $x_i^{c(k)}$ and can be computed via Eq. 18.4 (Kaveh and Talatahari 2009):

$$\begin{gathered} x_i^{(k+1,j)} = \alpha_2 x_i^{c(k)} + (1-\alpha_2)\left(\alpha_3 x_i^{gbest(k)} + (1-\alpha_3) x_i^{lbest(k,j)}\right) + \frac{r_j \alpha_1 (x_{\max} - x_{\min})}{k+1}, \\ i = 1, 2, \ldots, ng \\ j = 1, 2, \ldots, N, \end{gathered} \tag{18.4}$$

where r_j is a random number from a standard normal distribution which changes for each candidate; $x_{\max}$ and $x_{\min}$ are the upper and lower limits; α_1 is a parameter for limiting the size of the search space; $x_i^{lbest(k,j)}$ is the best position of the jth particle up to the iteration k; $x_i^{gbest(k)}$ is the best position among all candidates up to the iteration k; α_2 and α_3 are adjustable parameters controlling the influence of the global best and local best on the new position of the candidates, respectively.

Another reformation in the HBB–BC is that the SOM has been employed as an auxiliary tool to update the search space. Based on the principle of finite element method, SOM was introduced by Kaveh et al. (2008). The work principle of SOM is repetitive dividing the search space into sub-domains and employing optimization process into these sub-domains until a specified termination criteria (such as required accuracy) is satisfied and return the best solution.

The SOM mechanism can be calculated as the repetition of the following steps for definite times, nc, (in the stage k of the repetition):

- Calculating cross-sectional area bounds for each group.

If $x_i^{gbest(k_{SOM-1})}$ is the global best solution obtained from the previous stage for design variable i, then we have Eq. 18.5 (Kaveh and Talatahari 2009):

$$\begin{gathered} \begin{cases} x_{\min,i}^{(k_{SOM})} = x_i^{gbest(k_{SOM-1})} - \beta_1 \cdot \left(x_{\max,i}^{(k_{SOM-1})} - x_{\min,i}^{(k_{SOM-1})}\right) \geq x_{\min,i}^{(k_{SOM-1})} \\ x_{\max,i}^{(k_{SOM})} = x_i^{gbest(k_{SOM-1})} + \beta_1 \cdot \left(x_{\max,i}^{(k_{SOM-1})} - x_{\min,i}^{(k_{SOM-1})}\right) \leq x_{\max,i}^{(k_{SOM-1})} \end{cases}, \\ i = 1, 2, \ldots, ng \\ k_{SOM} = 2, \ldots, nc, \end{gathered} \tag{18.5}$$

where is an adjustable factor which determines the amount of the remaining search space; and $x_{\min,i}^{(k_{SOM})}$, $x_{\max,i}^{(k_{SOM})}$ are the minimum and the maximum allowable cross-sectional areas at the stage, respectively. In stage 1, the amounts of $x_{\min,i}^{(1)}$ and $x_{\max,i}^{(1)}$ are set according to Eq. 18.6 (Kaveh and Talatahari 2009):

$$x_{\min,i}^{(1)} = x_{\min}, \quad x_{\max,i}^{(1)} = x_{\max}, \quad i = 1, 2, \ldots, ng, \tag{18.6}$$

where $x_{\min}, i^1$ and $x_{\max}, i^1$ are the minimum and the maximum allowable cross-sectional areas at the stage 1.

- Determining the amount of increase in allowable cross-sectional areas via Eq. 18.7 (Kaveh and Talatahari 2009).

$$x_i^{*(k_{SOM})} = \frac{\left(x_{\max,i}^{(k_{SOM})} - x_{\min,i}^{(k_{SOM})}\right)}{\beta_2 - 1}, \quad i = 1, 2, \ldots, ng, \tag{18.7}$$

where $x_i^{*(k_{SOM})}$ is the amount of increase in allowable cross-sectional area; and β_2 is the number of permissible value of each group.

- Creating the series of the allowable cross-sectional areas.
The set of allowable cross-sectional areas for group i can be defined as Eq. 18.8 (Kaveh and Talatahari 2009):

$$x_{\min,i}^{(k_{SOM})}, \; x_{\min,i}^{(k_{SOM})} + x_i^{*(k_{SOM})}, \; \ldots, x_{\min,i}^{(k_{SOM})} + (\beta_2 - 1) \cdot x_i^{*(k_{SOM})} = x_{\max,i}^{(k_{SOM})}, \quad i = 1, 2, \ldots, ng. \tag{18.8}$$

- Determining the optimum solution of the stage k_{SOM}.
This is the last step and the stopping creation for SOM can be defined as Eq. 18.9 (Kaveh and Talatahari 2009):

$$x_i^{*(nc)} \leq x^*, \quad i = 1, 2, \ldots, ng, \tag{18.9}$$

where $x_i^{*(nc)}$ = the amount of accuracy rate of the last stage; and x^* = the amount of accuracy rate of the primary problem.

In addition to HBB–BC, another hybridization between the BB–BC algorithm and simulated annealing (SA) technique was recently proposed in Altomare et al. (2013). In this approach, the value of fitness function is further submitted to a local optimization by SA with a fast annealing schedule. This new hybrid method has been implemented to solve crystal structure problems. Compared with traditional SA algorithm, the hybridized algorithm showed better results in terms of computation time.

18.2.3.2 Improved BB–BC Algorithm

To improve the BB–BC performance, Hasançebi and Azad (2012) proposed two enhanced variants of the BB–BC algorithm, called modified BB–BC (MBB–BC) and exponential BB–BC algorithm (EBB–BC), respectively. In the new formulation, the normal random number (r) is changed by using any appropriate statistical distribution in order to eliminate the shortcomings of the standard formulation (e.g., big search dimensionality). Furthermore, to meet the discrete data requirements, the improved BB–BC algorithm employed the way of round-off instead of the real values to nearest integers representing the sequence number. As a result, the new generation for the next iteration in the big bang phase can be formulated as Eq. 18.10 (Hasançebi and Azad 2012):

$$x^{new} = x^c + round\left[\alpha \cdot N(0,1)_i^3 \frac{(x_{\max} - x_{\min})}{k}\right], \tag{18.10}$$

where x^c is the value of discrete design variable, $x_{\max}$ and $x_{\min}$ are its lower and upper bounds, respectively. In addition, the power of random number is set to 3 based on extensive numerical experiments. This reformulation is referred to as MBB–BC.

In a similar vein, Hasançebi and Azad (2012) also proposed an alternative approach called EBB–BC to deal with the discrete design problem where the use of an exponential distribution (E) in conjunction with the third power of random number as shown in Eq. 18.11.

$$x^{new} = x^c + round\left[\alpha \cdot E(\lambda = 1)_i^3 \frac{(x_{\max} - x_{\min})}{k}\right]. \tag{18.11}$$

The probability density function for an exponential distribution is given as Eq. 18.12 (Hasançebi and Azad 2012):

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}, \tag{18.12}$$

where λ is a real, positive constant. The mean and variance of the exponential distribution are given as $1/\lambda$ and $1/\lambda^2$, respectively.

Accordingly, if all the design variables in a new solution remain unchanged after applying Eq. 18.11, i.e., $x^{new} = x^c$, the generation process is iterated in the same way by decreasing the λ parameter of the exponential distribution by half each time, and this is repeated until a different solution is produced, i.e., $x^{new} \neq x^c$.

Hasançebi and Azad (2012) presented two numerical examples to investigate the performance of EBB–BC and MBB–BC. Compared with standard BB–BC, the improved variants gave the better results in terms of balancing between the exploration and exploitation characteristics of the algorithms. More recently, an upper bound strategy (UBS) with MBB–BC and EBB–BC was further integrated in Azad et al. (2013) for optimum design of steel frame structures. Computational results showed that the new effort significantly reduced the number of structural analyses.

Furthermore, to improve the convergence properties of the BB–BC, Alatas (2011) proposed a new methods called uniform big bang–chaotic big crunch (UBB–CBC) algorithm which involves two improved reformulation, i.e., an uniform population method to generate uniformly distributed random points in the big bang phase (called UBB), and the chaotic maps property to rapidly shrink those points to a single representative point in the big crunch phase (called CBC). Compared with benchmark functions, the performance of UBB–CBC showed superiority over the standard BB–BC algorithm.

18.2.3.3 Local Search-Based BB–BC Algorithm

As Kaveh and Talatahari (2009) concluded at the end of their study, the HBB–BC is worse than improved algorithms which have the extra local search ability. To fulfil this gap, Genç et al. (2010) introduced a local search move mechanism to BB–BC algorithm based on defining a possible improving direction to check neighbouring points.

In details, Genç et al. (2010) put the local search methods (i.e., expansion and contraction) between the original "banging" and "crunching" phases. The main objective is to modify the representative point with local directional moves, in order to easily attack the path going to optima and decrease the process time fro reaching the global minima. The direction vector can be formulated as Eq. 18.13 (Genç et al. 2010):

$$IV_1 = P(n) - P(n-1), \tag{18.13}$$

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.

To test the performance of the new algorithm, Genç et al. (2010) implemented it on the target tracking problem. The simulation results showed that the local search-based BB–BC algorithm outperformed the standard BB–BC algorithm in terms of data accuracy.

18.2.4 Representative BB–BC Application

According to the literature review, the main application of the BB–BC algorithm is in structural optimization. In general, there are three main groups of structural optimization applications (Sadollah et al. 2012; Hasançebi and Azad 2012): (1) sizing optimization; (2) shape optimization; and (3) topology optimization. In sizing optimization, it can further be divided into two subcategories: discrete and continuous. Hasançebi and Azad (2012) used MBB–BC and EBB–BC to solve the discrete sizing optimization, whereas Kaveh and Talatahari (2009) and (2010a) proposed the HBB–BC algorithm to solve problems with continuous domains.

18.2.4.1 Truss Optimization

Truss optimization is one of the most active branches of the continuous sizing optimization. The main objective for designing truss structures is to determine the optimum values for member cross-sectional areas (A_i) in order to minimize the structural weight (W), meanwhile, satisfy the inequality constraints that limit design variable sizes and structural responses.

In Kaveh and Talatahari (2009), the authors employed the HBB–BC method to address the above mentioned truss optimization problem. In their work, five truss structures optimization examples were presented, namely, a 25-bar spatial truss structure, a 72-bar spatial truss structure, a 120-bar dome shaped truss, a square on diagonal double-layer grid, and a 26-story-tower spatial truss. Compared with other CI techniques (e.g., GA, ACO, and PSO), the HBB–BC performed well in large size structures characterized by converging difficulty or easily getting trapped at a local optimum.

18.3 Conclusions

In summary, the BB–BC algorithm is a population-based CI algorithm that shares some similarities with evolutionary algorithms (Erol and Eksin 2006), such as randomly selected initialization and refinement of the value of fitness function according to the best fitted answers of the previous loop or loops (Kaveh and Farhoudi 2011). The core working principle of BB–BC is to transform a convergent solution to a chaotic state which is a new set of solutions (Erol and Eksin 2006). The leading advantages of BB–BC are its high convergence speed and the low computation time, together with its simplicity and capability of easy-to-implement (Desai and Prasad 2013).

With the rapid spreading of BB–BC, in addition to the representative applications detailed in this chapter, the BB–BC has also been successfully applied to a variety of optimization problems as outlined below:

- Automatic target tracking (Genç et al. 2010; Genç and Hocaoğlu 2008).
- Fuzzy system control (Kumbasar et al. 2008, 2011; Aliasghary et al. 2011).
- Layout optimization (Kaveh and Farhoudi 2011).
- Linear time invariant systems (Desai and Prasad 2013).
- Course timetabling (Jaradat and Ayob 2010).
- Power system (Sedighizadeh and Arzaghi-Haris 2011; Dincel and Genc 2012; Kucuktezcan and Gen 2012; Zandi et al. 2012).
- Structural engineering (Altomare et al. 2013; Azad et al. 2013; Tang et al. 2010; Camp 2007; Camp and Huq 2013).

Interested readers are referred to them as a starting point for a further exploration and exploitation of the BB–BC algorithm.

References

Alatas, B. (2011). Uniform big bang–chaotic big crunch optimization. *Communications in Nonlinear Science and Numerical Simulation, 16*, 3696–3703.

Aliasghary, M., Eksin, I., & Guzelkaya, M. (2011). Fuzzy-sliding model reference learning control of inverted pendulum with big bang–big crunch optimization method. In *11th International Conference on Intelligent Systems Design and Applications (ISDA)* (pp. 380–384). IEEE.

Altomare, A., Corriero, N., Cuocci, C., Moliterni, A., & Rizzi, R. (2013). The hybrid big bang–big crunch method for solving crystal structure from powder diffraction data. *Journal of Applied Crystallography, 46*, 779–787.

Azad, S. K., Hasançebi, O., & Azad, S. K. (2013). Upper bound strategy for metaheuristic based design optimization of steel frames. *Advances in Engineering Software, 57*, 19–32.

Bauer, W., & Westfall, G. D. (2011). *University physics with modern physics*. New York, USA: McGraw-Hill. ISBN 978-0-07-285736-8.

Camp, C. V. (2007). Design of space trusses using big bang–big crunch optimization. *Journal of Structural Engineering, 133*, 999–1008.

Camp, C. V., & Huq, F. (2013). CO_2 and cost optimization of reinforced concrete frames using a big bang–big crunch algorithm. *Engineering Structures, 48*, 363–372.

Desai, S. R., & Prasad, R. (2013). A novel order diminution of LTI systems using big bang–big crunch optimization and routh approximation. *Applied Mathematical Modelling*, *37*, 8016–8028. http://dx.doi.org/10.1016/j.apm.2013.02.052.

Dincel, E., & Genc, V. M. I. (2012, November 23–25). A power system stabilizer design by big bang–big crunch algorithm. In *IEEE International Conference on Control System, Computing and Engineering*, Penang, Malaysia (pp. 307–312). IEEE.

Erol, O. K., & Eksin, I. (2006). A new optimization method: Big bang–big crunch. *Advances in Engineering Software, 37*, 106–111.

Genç, H. M., & Hocaoğlu, A. K. (2008). Bearing-only target tracking based on big bang–big crunch algorithm. In *The Third International Multi-Conference on Information Technology* (pp. 229–233). IEEE.

Genç, H. M., Eksin, İ., & Erol, O. K. (2010, October 10–13). Big bang–big crunch optimization algorithm hybridized with local directional moves and application to target motion analysis problem. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Istanbul, Turkey (pp. 881–887). IEEE.

Hasançebi, O., & Azad, S. K. (2012). An exponential big bang–big crunch algorithm for discrete design optimization of steel frames. *Computers and Structures, 110–111*, 167–179.

Jaradat, G. M., & Ayob, M. (2010). Big bang–big crunch optimization algorithm to solve the course timetabling problem. In *10th International Conference on Intelligent Systems Design and Applications (ISDA)* (pp. 1448–1452). IEEE.

Kaveh, A., & Farhoudi, N. (2011). A unified approach to parameter selection in meta-heuristic algorithms for layout optimization. *Journal of Constructional Steel Research, 67*, 1453–1462.

Kaveh, A., & Talatahari, S. (2009). Size optimization of space trusses using big bang–big crunch algorithm. *Computers and Structures, 87*, 1129–1140.

Kaveh, A., & Talatahari, S. (2010a). A discrete big bang–big crunch algorithm for optimal design of skeletal structures. *Asian Journal of Civil Engineering (Building and Housing), 11*, 103–122.

Kaveh, A., & Talatahari, S. (2010b). Optimal design of Schwedler and ribbed domes via hybrid big bang–big crunch algorithm. *Journal of Constructional Steel Research, 66*, 412–419.

Kaveh, A., Farahmand, B. A., & Talatahari, S. (2008). Ant colony optimization for design of space trusses. *International Journal of Space Structure, 23*, 167–181.

Kucuktezcan, C. F., & Genc, V. M. I. (2012). Big bang–big crunch based optimal preventive control action on power systems. In *3rd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe)*, Berlin, Germany (pp. 1–4). IEEE.

Kumbasar, T., Yeşil, E., Eksin, İ., & Güzelkaya, M. (2008, March 12–14). Inverse fuzzy model control with online adaptation via big bang–big crunch optimization. In *3rd International Symposium on Communications, Control and Signal Processing*, Malta (pp. 697–702). IEEE.

Kumbasar, T., Eksin, I., Guzelkaya, M., & Yesil, E. (2011). Adaptive fuzzy model based inverse controller design using BB–BC optimization algorithm. *Expert Systems with Applications, 38*, 12356–12364.

Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2012). Mine blast algorithm for optimization of truss structures with discrete variables. *Computers and Structures, 102–103*, 49–63.

Scalzi, J. (2008). *The rough guide to the universe*. New York, USA: Rough Guides Ltd. ISBN 9781-84353-800-4.

Sedighizadeh, M., & Arzaghi-Haris, D. (2011). Optimal allocation and sizing of capacitors to minimize the distribution line loss and to improve the voltage profile using big bang–big crunch optimization. *International Review of Electrical Engineering, 6*, 2013–2019.

Tang, H., Zhou, J., Xue, S., & Xie, L. (2010). Big bang–big crunch optimization for parameter estimation in structural systems. *Mechanical Systems and Signal Processing, 24*, 2888–2897.

Zandi, Z., Afjei, E., & Sedighizadeh, M. (2012, Dec 2–5). Reactive power dispatch using big bang–big crunch optimization algorithm for voltage stability enhancement. In *IEEE International Conference on Power and Energy (PECon)*, Kota Kinabalu Sabah, Malaysia (pp. 239–244). IEEE.