An overview of Algorithms on Integer Calculation

Abstract

This work proposes a new meta-mathematical method called arithmetic optimization algorithm that uses the distributive behavior of the main arithmetic operators in mathematics, including division, subtraction, and addition for now the number of distinct entries in the multiplication table n. Therefore, there is some interest in algorithms for calculating M (n) exactly or as an approximation. We compare several algorithms for exact calculation of M (n) and come up with a new algorithm with suborder execution time. We also present two Monte Carlo algorithms to approximate M (n). We give exact calculation results for values of n to 230 and compare our experimental results with Ford's order of magnitude results. Experimental results show that INTEGERS provides very promising results in solving difficult optimization problems compared to 11 other well-known optimization algorithms.

Keyword: Integer, calculation, algorithm

1. Introduction

In unpublished work since 2012, the first two authors revisited the problem, extending the calculation to n = 25, and discovering a Monte Carlo estimate for larger n. A few years later, the fourth author discovered some new algorithms to compute M (n) exactly. We present both the exact algorithm and the Monte Carlo algorithm. It is useful to distinguish between an algorithm that evaluates M(n) at a given integer n and a tabulation algorithm M(k) for any integer k in the interval [1; n], which we can simply call the M(n) tab".

To confirm the numerical results, we used Algorithm 1 for various values of n, including n = 2k 1 for k = 1; 2; 50. The known precision evaluation/tabulation algorithms are too slow to exceed n = 230, so for larger n, approximation methods need to be resorted to. We introduce two Monte Carlo algorithms, which we call Bernoulli and productive, for reasons that will be clearly presented in the next section. In each case, we avoid the problem of factoring large integers by using Bach's algorithm to generate random integers in digitized form. The speed of Monte Carlo algorithms mainly depends on the time it takes to check for primality6 of large integers, which can be done much faster than factoring integers have the same size.

We assume that arithmetic operations, memory accesses, and other basic operations take units of time. We are counting the space in bits and not including any space used to store the output. Sieve of Eratosthenes Algorithms for evaluating M(n) are the same as sieve of Eratosthenes, the simplest implementation involving, for each 1 < k n1 = 2, remove multiples of k from O(n) time and space and find all primes up to n. There is an extensive literature, both practical and theoretical, that deals with improvements and variations of this sieve. Here, we highlight relevant aspects for the calculation of M (n). In practice, one may be limited by a space constraint; reduce the space used by an algorithm that can turn impractical calculations into actionable ones. Using this idea, the space limit can be reduced to O (n1 = 2) with a direct segmentation of the interval [1; Not]. Helot further reduces the space required by using a Diophantine approximation to predict which integers less than n1 = 2 have multiples in a given subinterval. This prediction process makes it possible to screen intervals of size O (n1 = 3 (log n) 5 = 3) without wasting time on asymptotes.

2. Background

For more details and convergence analysis of spatial branching methods in particular as well as to get an overview of overall optimization in general. Also common today are mixed integer solvers such as results [1, 2, 3].

The optimization process finds the optimal decision variables of a function or problem by minimizing or maximizing its objective function. In general, optimization and real-world problems have limitations of non-linearity, high computational time, non-convexity, and large search space [5, 6], to making them difficult to solve. Hypersimulation optimization algorithms have two important search strategies:

(1) exploration/diversification and

(2) mining/enhancement [7,8].

Exploration is the ability to explore the research space on a global scale. This capability involves avoiding local optimas and resolving local optima traps. In contrast, mining is the ability to discover nearby promising solutions to improve their quality locally [9].

Excellent performance of an algorithm requires a good balance between these two strategies [10–12].

All population-based algorithms use these features but with different operators and mechanisms. A popular classification of based metaheuristics inspired by evolutionary algorithms, swarm intelligence algorithms, physics-based methods, and human-based methods [13,14].

Evolutionary algorithms simulate the habits of natural evolution and use operators driven by biological behaviors such as hybridization and mutation. One common evolutionary algorithm is the Genetic Algorithm (GA), which is motivated by Darwin's evolutionary ideas. Common methods in this group include evolutionary programming [15], differential evolution [16], and evolutionary strategy [17]. Herd intelligence algorithms are another group of hypersimulators, simulating the behavior of animals in motion or in groups of prey [18,19]. The main feature of this group is to share information about the organism of all animals through the optimization course. Common methods in this group include Krill Swarm Algorithm [20], Salp Swarm Algorithm [20], Symbiotic Search [21], Cosine Algorithm [22], and Dolphin Locator [21] 23]. Another group of optimization algorithms are physics-based methods. This group is rooted in the physical laws of real life and generally describes the communication of research solutions based on control rules embedded in physical methods. The most commonly used algorithms in this group are simulation annealing [24], gravity search algorithm [25], multiverse optimizer [26], and loaded system search [27]. . The final group of optimizations are people-based methods, driven by human cooperation and human behavior in the community. One of the most used algorithms in this group is the competitive imperial algorithm [28], which is driven by human socio-political development. Another algorithm in this group is the teaching-learning-based optimization algorithm [29]. The theoretical studies published in the literature can be classified into three parts: modifying existing algorithms, combining different algorithms, and proposing new algorithms. These three areas are very dynamic with a large number of algorithms and applications. The reason why researchers do not use a single algorithm is that there is no single optimization algorithm to solve all the optimization problems according to the Free Lunch Theorem [30].

3. Integer Calculation Survey

Two-level optimization is an area of mathematical programming in which certain variables are constrained to be the solution of another optimization problem. As a result, two-level optimization can model hierarchical decision-making processes. This is interesting for modeling real-world problems, but it also makes the resulting optimization models difficult to solve theoretically and practically. Scientific interest in two-level computational optimization has increased dramatically over the past decade and continues to grow. Regardless of whether the two-level problem itself contains integer variables or not, many advanced solution approaches to techniques using two-level optimization come from mixed integer programming. These techniques include branching methods, slicing lines and thus branching processing methods, or problem-specific analysis methods. In this research paper, we consider suitable two-level approaches that exploit these mixed integer programming techniques to solve two-level optimization problems. To achieve this, we first consider two-level problems with convexity or, in particular, lower-order linear problems. The solving methods discussed in this field come from original work from the 1980s but on the other hand, are still being actively studied to this day. Second, we consider modern algorithmic approaches to solving two-level problems of mixed integers containing lower-order integral constraints. In addition, we also briefly discuss the domain of nonlinear two-level mixed number problems. Third, we pay attention to some more specific areas such as pricing or prohibition models that actually contain bilinear and therefore non-convex aspects. Finally, we put together a list of open questions in the areas of algorithmic and computational optimization on two levels, which could lead to interesting future research that will advance this fascinating field of research. lead and positive this further. We use the nomenclature where a two-level (1) problem is called a "ULLL problem" where UL and LL can be LP, QP, MILP, MIQP, etc. whether the superior/subordinate problem is linear, quadratic, linear mixed, quadratic integer mixture, etc. program in both leader and follower variables. If a two-level specific specification is not required, we also use a shorter BOM, and the problem is for example a two-level LP, if both levels are LPs. Most of the time, we will consider the optimistic version of the two-level problem as given in (1). In this case, the leader also optimizes the lower-level result ∈() if the lower-level solution set () is not a singleton. The introductory formula was first used by Bracken and McGill (1973) in the military context of applying a minimal cost weapon mix. Another very old discussion of multilevel, or specifically two-level problems, can be found in Candler and Norton (1977). Over the years, two-level optimization has been recognized as an important modeling tool as it allows the formalization of hierarchical decision-making processes that often appear in application areas such as energy , security or income management. We postpone the discussion of the selected application materials to the following sections. The ability to model hierarchical decision-making processes also makes two-level optimization problems notoriously difficult to solve. Therefore, efficient algorithms, i.e. in polynomial time, cannot be expected unless P = NP. It also makes it difficult to develop algorithms that solve on the one hand, "enable" enumeration-based algorithms like branchandbound on the other. In recent years and decades, the development of algorithms for solving two-level optimization problems depends strongly on the structure and properties of the lower-level problem as well as on the coupling between the higher-order and the lower-level problems. higher and lower level. For example, solving techniques vary widely depending on whether the sub-problem is continuous and convex or it is non-convex, for example due to the presence of integer variables. In this investigation, we focus on algorithmic techniques to actually solve two-level problems. In particular, we discuss mixed linear or nonlinear optimization techniques applied in the field of two-level optimization. These basic and well-studied techniques include fork (Land and Doig, 1960) or shear (Kelley, 1960) designs as well as decomposition techniques such as as a Benders decomposition (general) (Benders, 1962; Geo ffrion, 1972); see books by Conforti et al. (2014); Junger et al. (2010); Wolsey (1998) for a comprehensive overview of linear mixed number programming techniques. In addition, specific techniques of nonlinear mixed integer programming such as external approximation (Bonami et al., 2008; Duran and Grossmann, 1986; Fletcher and Leyffer, 1994) or spatial branching (Horst et al. Tuy, 2013) is mentioned; see Belotti et al. (two thousand and thirteen); Lee (2012) for recent insights into nonlinear optimization of mixed integers. For the more theoretical aspects of two-level optimization, we refer to Dempe (2002) and the references therein.

Typically, population-based algorithms begin their improvement process with a randomly generated set of candidate solutions. This generated solution set is improved by an incremental set of optimization rules and is evaluated against a specific objective function iteratively; this is the essence of optimization methods. Since set-based algorithms search for optimal solutions to optimization problems by a random method, obtaining a solution in one run is not guaranteed. The probability of getting the overall optimal solution, for the given problem, is increased by a sufficient number of random solutions and optimization iterations.

Despite the differences between metaheuristic algorithms in the field of population-based optimization methods, the optimization process consists of two main stages: exploration and mining. The first deals with extensive coverage of the search space using the algorithm's search agents to avoid local solutions. The latter is to improve the accuracy of the solutions obtained during the exploration phase. We use these simple operators as a mathematical optimization to determine the best element that conforms to specific criteria from a set of candidate alternatives.

Optimization problems arise in all quantitative disciplines, from engineering, economics, and computer science to operations and industrial research, and improvements in solving techniques have rekindled the love for math of all ages. The main inspiration of the integer proposal came from the use of arithmetic operators to solve arithmetic problems.

4. Integers Computational Complexity and Algorithms

Note that the proposed integer computational complexity is essentially based on three factors: the initialization process, matching function evaluation, and updating solutions. The complexity of the initialization is O(N) where N represents the size of the population. The complexity of the fitness function depends on the problem, so we do not discuss it here.

Finally, the complexity of updating solutions is $O(M \times N) + O(M \times N \times L)$ where M is the number of iterations and L is the number of parameters in the problem (dimensions). Therefore, the computational complexity of the proposed integer is $O(N \times (ML + 1))$.

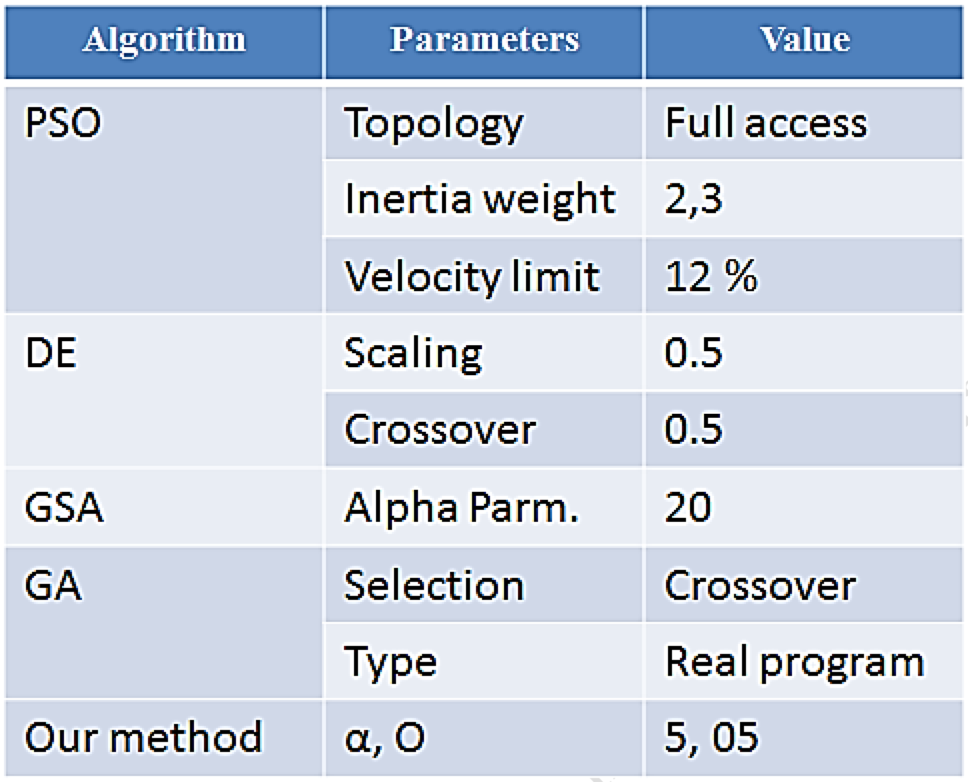
In the next section, various benchmarking functions and actual optimization problems used to validate and validate the performance of integer are proposed to solve optimization problems. ving arithmetic problems.

The performance of the proposed integer algorithm is tested on some test functions and five engineering design problems. The results are compared with the following algorithms:

- Genetic Algorithm (GA)
- Particle Swarm Optimization (PSO)
- Gravitational Search Algorithm (GSA)
- Differential Evolution (DE)

To get a fair comparison, the algorithms considered were implemented using the same number of iterations and the same population size of the GEO 10,000, respectively, so the number of The functional rating is 12,000. The values used for the main control parameters of the comparison algorithms can be seen in Table 1.

At the beginning of this section, we examine the impact of changing the value of the integer‘s parameters on its performance. The different scenarios are considered as a function of the values of the parameters (O and) of integer.



Algorithm	Parameters	Value
PSO	Topology	Full access
	Inertia weight	2,3
	Velocity limit	12 %
DE	Scaling	0.5
	Crossover	0.5
GSA	Alpha Parm.	20
GA	Selection	Crossover
	Type	Real program
Our method	α, O	5, 05

Table1. Parameter values for the compare algorithms

These parameters are rated at Statistical results were obtained for each of the thirteen reference functions used. From these results, we can see that the fifth scenario (i.e. O = 0.5 and = 5), among all tested functions, has the better result; followed by the sixth and fourth scenarios, specifying the second and third rankings, respectively. The performance of the proposed integer algorithm is tested for the effects of dimensions, as shown in Table 8. This test is a standard test used in the literature on test functions. check the optimization benchmark, which can show the effect of the dimensions on the performance as a result of integer to demonstrate its ability not only for small size problems but also for large size problems.

5.1 Real-world applications

This Part solves five engineering design problems using the proposed algorithm: welding beam design problem, tension/compression spring design problem, normal design problem. Under pressure, 3 bar farm design problem and speed reduction problem. To solve these problems, a set of 30 solutions and 500 iterations was used in each implementation. The results obtained were compared with several similar techniques published in the literature. The following subsections present the results of the proposed Integer compared with the results of modern methods. In this paper, optimization problems with common constraints and constraints are selected to test the efficiency of the proposed integer.

Figure 1 also confirms the superiority of the integer algorithm based on the average fit values of the test functions.

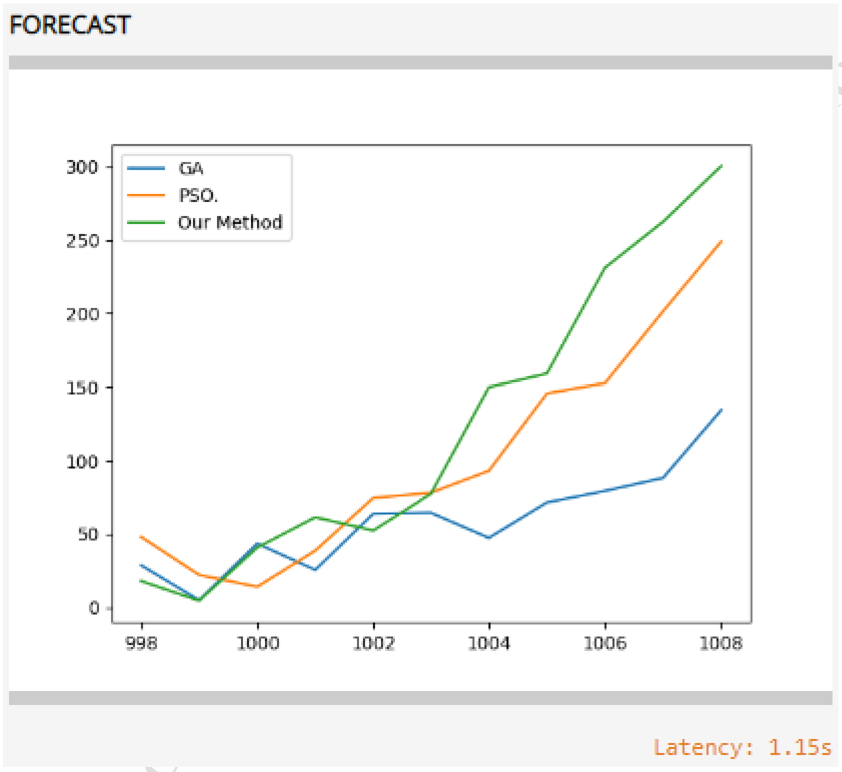
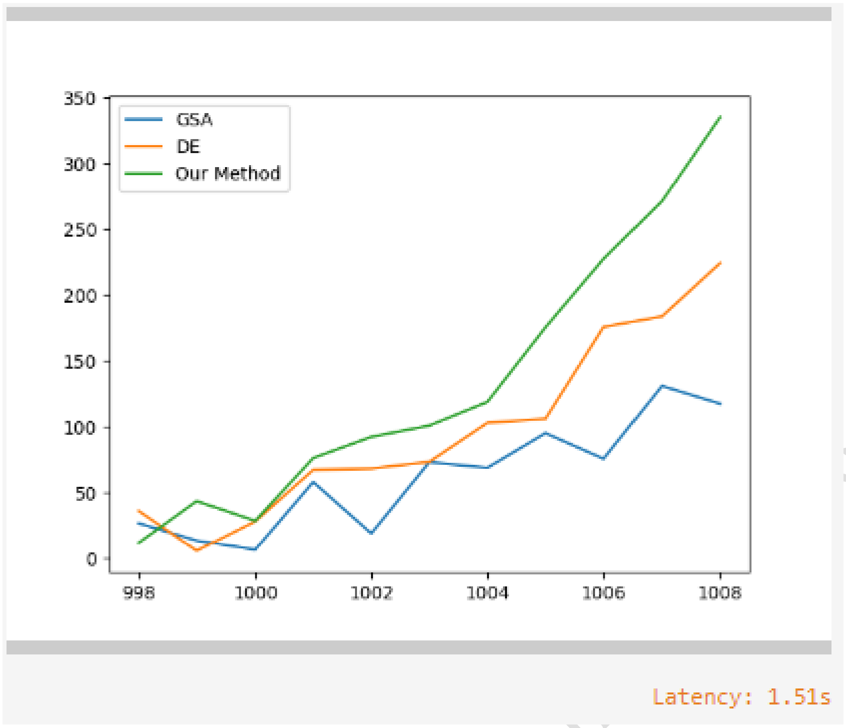


Figure 1 the superiority of the integer algorithms (PSO, GA and Our method)

Furthermore, it is clear from Figures 2 that integer does not confer a clear advantage in the first iterations on GSA, DE and our method.



Latency: 1.51s

Figures 2 that integer iterations on GSA, DE and our method

One of the reasons for this phenomenon is that integer distributes the locations of solutions to different local search areas instead of accumulating all locations in a local area based on good solutions. most available. However, the delivery mechanism, as mentioned earlier, increases the overall research capacity of integer. An integer can significantly avoid staying in local search areas and have competitive experience in finding global search regions.

Incorporating integer variables into the model is known to make problems harder to solve. It is often easier to design algorithms that are proven correct in solving two-level problems if all the bound variables are integers. This is why there are more methods for the two-level problems than there are published for mixed integers, i.e. the integer calculation case. In this survey, we mainly discuss the branching method as well as the branching method for solving two-level problems. If one compares the abundance of shear planes used in the field of one-level mixed integer optimization with the number of known valid inequalities for two-level optimization, it is clear that is that many branching and slicing algorithms will benefit from a particular set that is larger than the two-level cut. The whole field of solver engineering is almost completely unexplored in two-level optimization, while the performance of advanced system solvers depends heavily on it.

Conclusion

In addition, other improved versions of the proposed integer can be proposed to solve optimization problems with binary, discrete, and multiple objectives respectively. Stealth, interruption, mutation, and opposition-based learning can be combined with an integer to improve its performance. The whole algorithm can be combined with other random components, including methods of local search or global search, in the field of optimization to improve its performance. Finally, the study of integer usage in other disciplines will be a valuable contribution, such as in neural networks, image processing applications, feature selection, task planning in cloud computing, data and text mining applications, big data applications, signal reduction, resource management applications, smart home applications, networking applications, industrial and engineering applications , other benchmarking functions, other real-world problems.

References

1. Achterberg, T., 2019. What's new in Gurobi 9.0. In: Webinar Talk. URL: https://www. gurobi.com/wp-content/uploads/2019/12/Gurobi-9.0-Overview-Webinar-Slides-1.pdf.

2. Ambrosius, M., Grimm, V., Kleinert, T., Liers, F., Schmidt, M., Zöttl, G., 2020. En-dogenous price zones and investment incentives in electricity markets: An appli-cation of multilevel optimization with graph partitioning. Energy Economics 92. doi:10.1016/j.eneco.2020.104879.

3. J. Zhang, M. Xiao, L. Gao, Q. Pan, Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems, Appl. Math. Model. 63 (2018) 464–490.

4. W. Zhao, C. Du, S. Jiang, An adaptive multiscale approach for identifying multiple flaws based on xfem and a discrete artificial fish swarm algorithm, Comput. Methods Appl. Mech. Engrg. 339 (2018) 341–357.

5. L. Abualigah, Multi-verse optimizer algorithm: A comprehensive survey of its results, variants and applications, Neural Comput. Appl. (2020) 1–26.

6. V.V. de Melo, W. Banzhaf, Drone squadron optimization: a novel self-adaptive algorithm for global numerical optimization, Neural Comput. Appl. 30 (10) (2018) 3117–3144.

7. L. Abualigah, A. Diabat, S. Mirjalili et al. Computer Methods in Applied Mechanics and Engineering 376 (2021) 113609

8. L. Abualigah, A. Diabat, A comprehensive survey of the grasshopper optimization algorithm: results, variants, and applications, Neural Comput. Appl. (2020) 1–24.

9. L. Abualigah, A. Diabat, Z.W. Geem, A comprehensive survey of the harmony search algorithm in clustering applications, Appl. Sci. 10 (11) (2020) 3827.

10. L. Abualigah, Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications, Neural Comput. Appl. (2020) 1–24.

11. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, Knowl.-Based Syst. 191 (2020) 105190.

12. Sadollah, H. Sayyaadi, H.M. Lee, J.H. Kim, et al., Mine blast harmony search: a new hybrid optimization method for improving exploration and exploitation capabilities, Appl. Soft Comput. 68 (2018) 548–564.
13. S. Gholizadeh, M. Danesh, C. Gheyratmand, A new newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames, Comput. Struct. 234 (2020) 106250.
14. N.A. Kallioras, N.D. Lagaros, D.N. Avtzis, Pity beetle algorithm–a new metaheuristic inspired by the behavior of bark beetles, Adv. Eng. Softw. 121 (2018) 147–166.
15. L. Abualigah, M. Shehab, M. Alshinwan, H. Alabool, Salp swarm algorithm: a comprehensive survey, Neural Comput. Appl. (2019) 1–21.
16. L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution.
17. R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
18. N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), Evol. Comput. 11 (1) (2003) 1–18.
19. A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4831–4845.
20. L. Abualigah, M. Shehab, M. Alshinwan, S. Mirjalili, M. Abd Elaziz, Ant lion optimizer: A comprehensive survey of its variants and applications, Arch. Comput. Methods Eng. (2020).
21. S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
22. M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.
23. S. Mirjalili, Sca: a sine cosine algorithm for solving optimization problems, Knowl.-based Syst. 96 (2016) 120–133.
24. Kaveh, N. Farhoudi, A new optimization method: Dolphin echolocation, Adv. Eng. Softw. 59 (2013) 53–70.
25. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, science 220 (4598) (1983) 671–680.
26. E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, Gsa: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.
27. S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2) (2016) 495–513.
28. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mech. 213 (3–4) (2010) 267–289.
29. E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition,in: 2007 IEEE Congress on Evolutionary Computation, Ieee, 2007, pp. 4661–4667.
30. R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput. Aided Des. 43 (3) (2011) 303–315.