Structures 29 (2021) 107–128

Contents lists available at ScienceDirect

Structures

journal homepage: www.elsevier.com/locate/structures

Improved Shuffled Jaya algorithm for sizing optimization of skeletal structures with discrete variables

Ali Kaveh *, Seyed Milad Hosseini, Ataollah Zaerreza

School of Civil Engineering, Iran University of Science and Technology, P.O. Box 16846-13114, Iran

ARTICLE INFO

Keywords:
Metaheuristic algorithms
Improved Shuffled based Jaya (IS-Jaya) algorithm
Structural optimization
Discrete optimization of skeletal structures

ABSTRACT

Jaya algorithm is a simple and efficient population-based metaheuristic algorithm. Besides its simplicity, it has free from any algorithm-specific parameters. Although it has these advantages, the Jaya algorithm suffers from some shortcomings including unwanted premature convergence and the possibility of being trapped in local minima due to insufficient population diversity. To alleviate these handicaps, this paper proposes an Improved Shuffled based Jaya (IS-Jaya) algorithm. The proposed optimization method uses the concept of shuffling process to gain superior exploration capability in the search mechanism. A mechanism that causes to escape from local minima is also incorporated into the original Jaya algorithm. The efficiency of the IS-Jaya algorithm is tested on discrete optimization problems and compared to those of Jaya algorithm, self-adaptive multi-population-based Jaya (SAMP-Jaya), and some other state-of-art optimization methods. Optimization results show that the proposed optimization method can be an effective tool for solving discrete size optimization of skeletal structures.

1. Introduction

The process of finding the best or most efficient (a) size of structural elements (size optimization), (b) form for the structure (shape optimization), and (c) connectivity between structural elements (topology optimization) are called structural optimization. During this process, some constraints must be fulfilled. For instance, size optimization aims to minimize the weight of the structure while satisfying certain constraints on displacements, stress limitations, or both of them simultaneously. These challenging optimization problems have achieved significant attention from numerous researchers in recent past decades [1].

An appropriate optimizer should be taken into consideration to solve the optimization problems. In general, there are two optimization approaches namely deterministic and approximate to find an optimum solution. The main disadvantage of the first approach is that it lies in high computational complexity so that only limited types of optimization problems can be solved by this approach. In contrary, metaheuristics as probabilistic solvers, which belong to the latter approach, are capable of avoiding local minima and accelerating convergence as well. Structural optimization problems due to the complexity of their search space, cannot be solved easily using deterministic methods. Unlike deterministic ones, metaheuristic methods are capable to solve these categories of optimization problems at an affordable computational time with acceptable precision. Therefore, the metaheuristic optimization technique, as a robust alternative to the deterministic techniques, has been recently employed for solving structural optimization problems due to their capability of exploring and find promising solutions at an affordable time.

Literature reveals that different metaheuristic algorithms with different inspiration sources have been developed one after another in recent years. Some of the well-established of them can refer to: Genetic algorithm (GA) introduced by Holland [2], Simulated annealing (SA) introduced by Kirkpatrick et al. [3], Shuffled Complex Evolution (SCE) presented by Duan et al. [4], Particle Swarm Optimization (PSO) introduced by Kennedy and Eberhart [5], Ant Colony Optimization (ACO) developed by Dorigo et al. [6], Artificial Bee Colony (ABC) introduced by Karaboga [7], Shuffled Frog-Leaping Algorithm (SFLA) developed by Eusuff et al. [8], Cuckoo Search (CS) presented by Yang and Deb [9], Differential Evolution (DE) developed by Das and Suganthan [10], Teaching–Learning-based Optimization (TLBO) proposed by Rao et al. [11], Bat Algorithm (BA) developed by Yang and Gandomi [12], and Grey Wolf Optimizer (GWO) proposed by Mirjalili et al. [13]. Besides serious efforts in all advanced fields dealing with metaheuristics and their applications, the first author and his students have developed numerous frameworks for metaheuristics. Some of them include

* Corresponding author.
E-mail addresses: alikaveh@iust.ac.ir (A. Kaveh), hosseini_milad@civileng.iust.ac.ir (S.M. Hosseini), a_zaerreza@civileng.iust.ac.ir (A. Zaerreza).

https://doi.org/10.1016/j.istruc.2020.11.008
Received 2 June 2020; Received in revised form 5 November 2020; Accepted 6 November 2020
Available online 20 November 2020

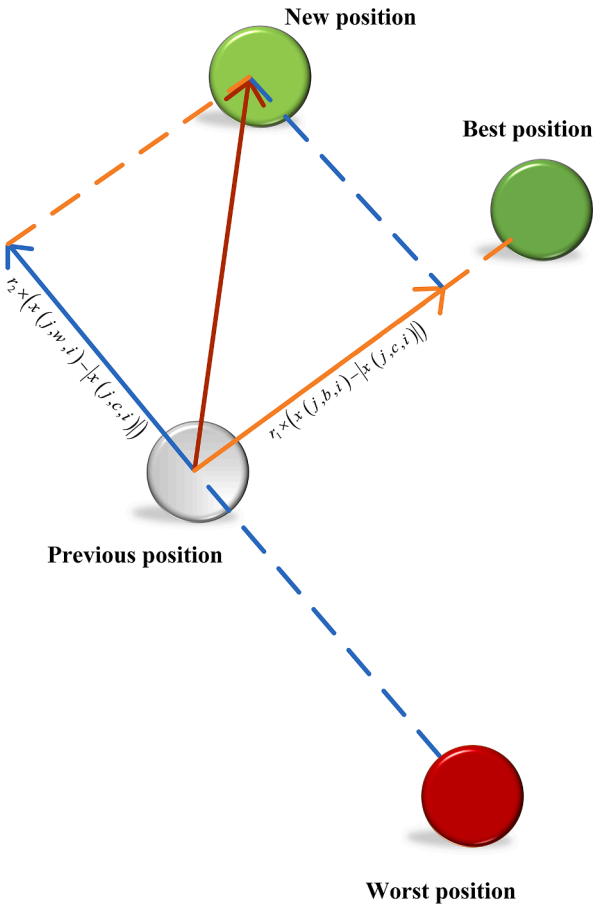

Fig. 1. A schematic of generating a new position in the Jaya algorithm.

Charged System Search (CSS) [14], Ray Optimization (RO) [15], Dolphin Echolocation (DE) [16], Colliding Bodies Optimization (CBO) [17], Water Evaporation Optimization (WEO) [18], Thermal Exchange Optimization (TEO) [19], Shuffled Shepherd Optimization Algorithm (SSOA) [20], and Water Strider Algorithm (WSA) [21].

Multi-population techniques have been recently used for enhancing the search mechanism of optimization methods. In this technique, the entire population is partitioned into several subpopulations, each of which is allowed to modify the solutions in each group independently. Dividing the whole population into subpopulations distributes the solutions over the entire search space instead of focusing on a certain region. As a result, since the subpopulations can be located in a different region of search space, this technique causes the algorithm to explore different areas of the search space simultaneously. Furthermore, many metaheuristic algorithms can be quickly and easily embedded into multi-populations methods. Both of these reasons reveal why multi-population methods are effective and popular [22]. In multi-population methods at the end of each iteration, the subpopulations interact with each other via merging. The size of each subpopulation can be either changeable adaptively or fixed throughout the optimization process. Therefore, the number of subpopulations is changing (increase or decrease) in the first case or are fixed in the latter case. Self-Adaptive Multi-Population based Jaya (SAMP-Jaya) algorithm is an example of a multi-population metaheuristic in which the subpopulation numbers continuously changes during the search process [23], while SCE, SFLA, and SSOA are instances of multi-populations optimization methods with a fixed number of subpopulations. The advantage of using fixed subpopulation numbers is that it can be implemented simply. In this regard, the fixed number of subpopulations is only determined by the researcher's experience.

In this paper, we have focused on a powerful and simple optimizer namely the Jaya algorithm developed by Rao [24]. Jaya and its variants have been applied to many optimization problems in different fields. Here, we are briefly summarizing some improvements and applications of Jaya, which have been recently developed. Rao and Saroj [23] proposed a Self-Adaptive Multi-Population based Jaya (SAMP-Jaya) algorithm. Farah and Belazi [25] presented a novel Chaotic Jaya algorithm (C-Jaya) to boost the search capability of the algorithm. Ghavidel et al. [26] proposed an efficient improved hybrid Jaya algorithm based on Time-Varying Acceleration Coefficients (TVACs) and the learning phase of TLBO. Wang and Huang [27] proposed a novel Elite Opposition-based Jaya (EO-Jaya) for parameter identification of photovoltaic (PV) cell models. Aslan et al. [28] presented a novel binary optimizer (JayaX) based on the Jaya algorithm and logic operator. Degertekin et al. [29] developed a novel formulation of the powerful Jaya algorithm namely Discrete Advanced Jaya Algorithm (DAJA) for discrete sizing, layout, and topology optimization of truss structures. Ding et al. [30] presented a new improvement of the Jaya algorithm for structural damage identification with the modified objective function based on sparse regularization and Bayesian inference. Migallón et al. [31] presented Jaya-based parallel algorithms to efficiently exploit cluster computing platforms. Nayak et al. [32] added the concept of mutation to the standard version of Jaya optimization (MJaya) for increasing the global search ability of the agents by providing additional diversity. In 2019, Rao and Saroj [33] proposed an Elitist-based Self-Adaptive Multi-Population (SAMPE) Jaya algorithm to improve the search mechanism of standard Jaya using subpopulation search scheme with elitism. Ingle and Jatoth [34] proposed an efficient Jaya algorithm with Lévy flight for training the Neural Network (NN) based equalizers.

Jaya algorithm with a simple formulation and powerful search mechanism attempts to move toward the best solution and tries to away from the worst solution in the search space. This simple search mechanism describes the main concept of the Jaya algorithm. On the other hand, there are no algorithm-specific parameters in the algorithm so that it is free from the initial guess of algorithm parameters, which leads to lower computational cost. Therefore, simplicity, efficiency, and having no algorithmic-specific parameters can be considered as advantages of the Jaya algorithm. However, it severely suffers from unwanted premature convergence and can be trapped in local minima due to weak exploration ability and insufficient diversity of the population. In order to alleviate these drawbacks, in the present study the concept of shuffling process and a mechanism to escape from local minima have been incorporated simultaneously into the standard Jaya algorithm, denoted as Improved Shuffled based Jaya (IS-Jaya) algorithm. To this end, first, the whole population is sorted based on the quality of solutions. Then, the population is divided into fixed subpopulations numbers, denoted as communities. In the IS-Jaya algorithm, the communities are generated through the process of shuffling. Each community is permitted to evolve independently by finding promising solutions based on the standard Jaya algorithm. This strategy makes to improve the solution by sharing the information independently acquired by each community. In order to avoid falling into the local optimal and improve the performance of the original Jaya algorithm, a mechanism is utilized to escape from local optima in each community. Finally, the communities share their information via merging to avoid unwanted premature convergence and also maintain population diversity. To assess the performance of the IS-Jaya algorithm, four benchmark design examples with discrete variables are investigated. The reason for selecting structural optimization problems with discrete design variables is because of their nonlinear programming with multi-nonlinear constraints [35], NP-hard [36,37], discontinuous, and non-convex search spaces with several local optima. In all examined examples, the results of the IS-Jaya algorithm are compared with the results obtained by Jaya, SAMP-Jaya, and other existing optimization methods published in the literature.

The rest of this paper is organized into the following sections: In Section 2 and 3, a brief overview of the Jaya and SAMP-Jaya algorithms

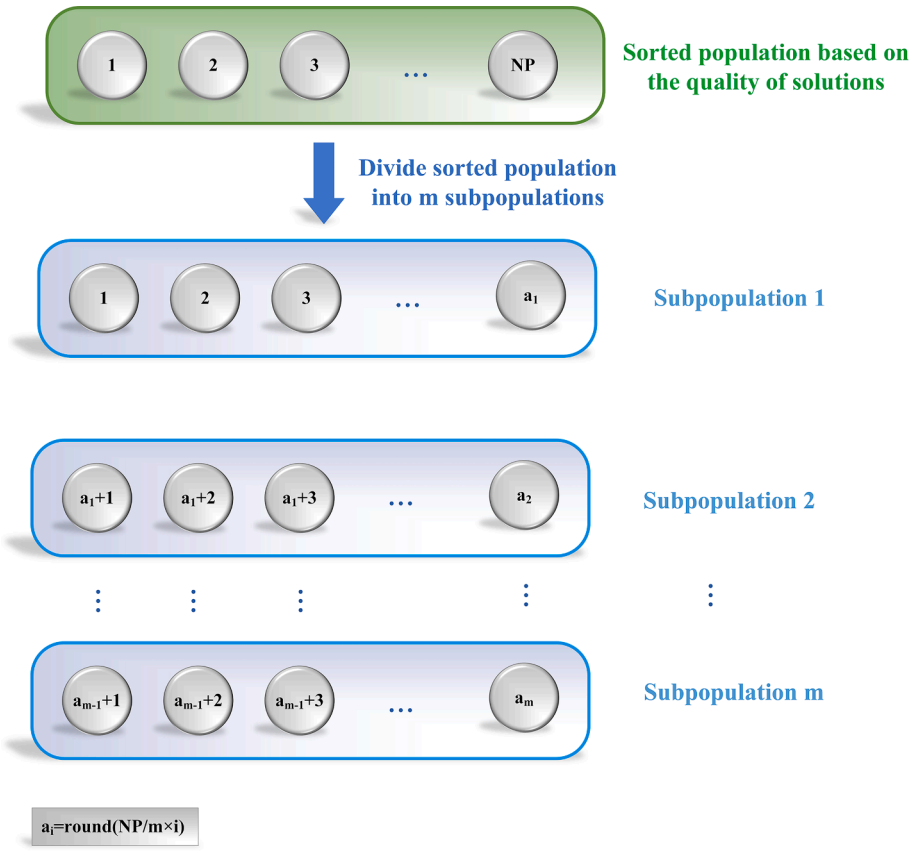

Fig. 2. A schematic of population partitioning in the SAMP-Jaya algorithm.

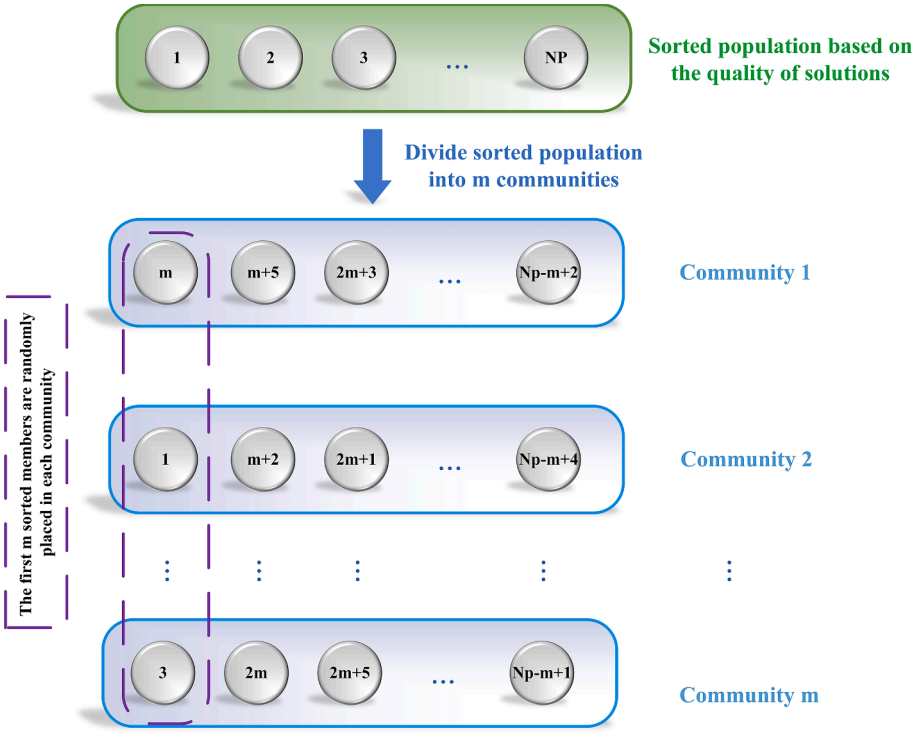

Fig. 3. A schematic of population partitioning in IS-Jaya.

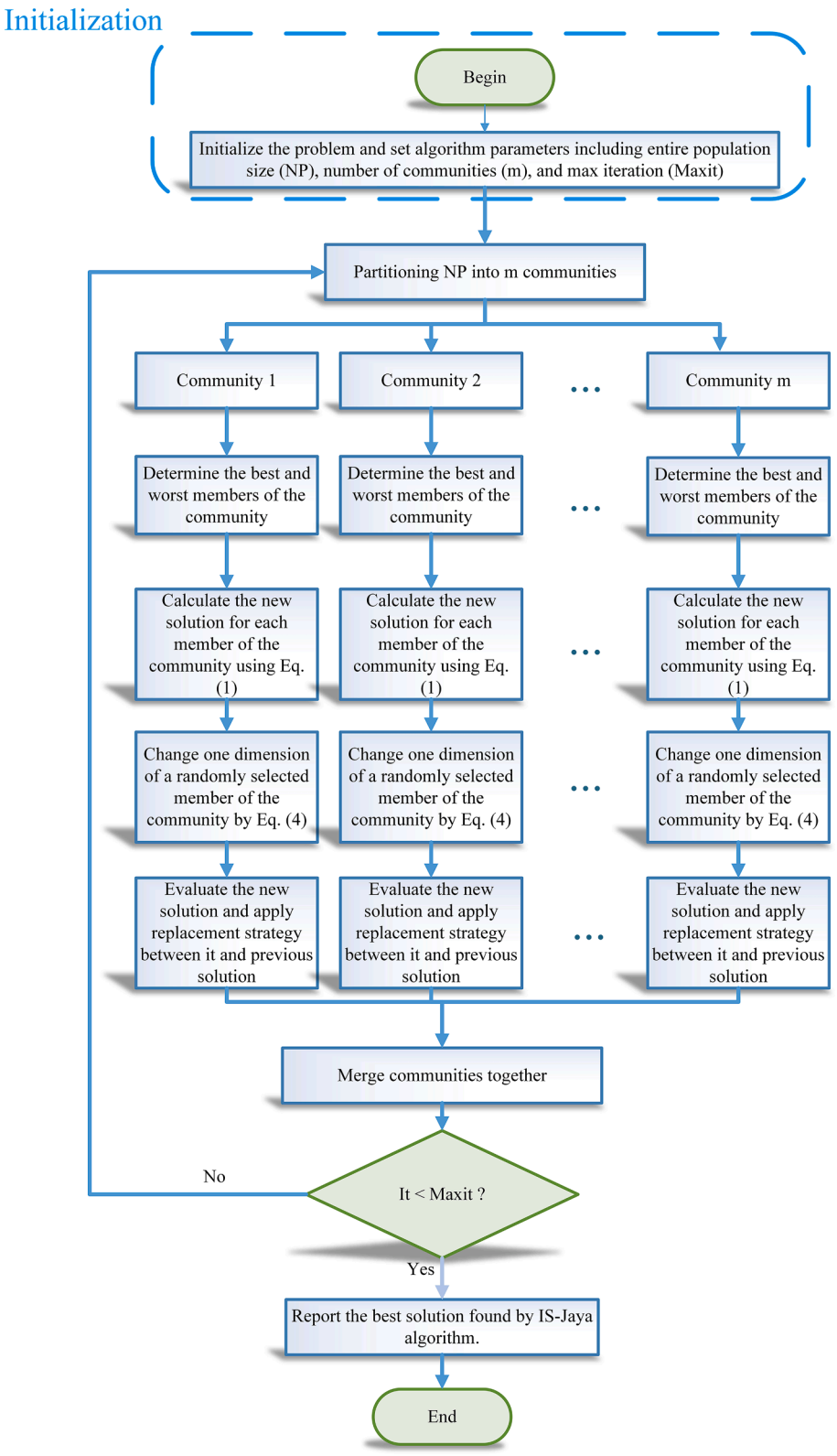

Fig. 4. Flowchart of the IS-Jaya algorithm.

are presented. In Section 4, the improved version of the Jaya algorithm (i.e. IS-Jaya) is introduced. In Section 5, the definition of structural optimization with discrete design variables is described. Section 6 uses four benchmark design examples to compare the IS-Jaya algorithm against Jaya, SAMP-Jaya, and some other state-of-art optimization methods. The conclusions are finally derived in Section 7.

2. Jaya algorithm

Jaya algorithm is the newly developed population-based meta-heuristic, which has been proposed by Rao in 2016 [24]. Jaya is a Sanskrit word which means victory or triumph. Efficiency evaluation and robustness of this optimization method have been demonstrated in

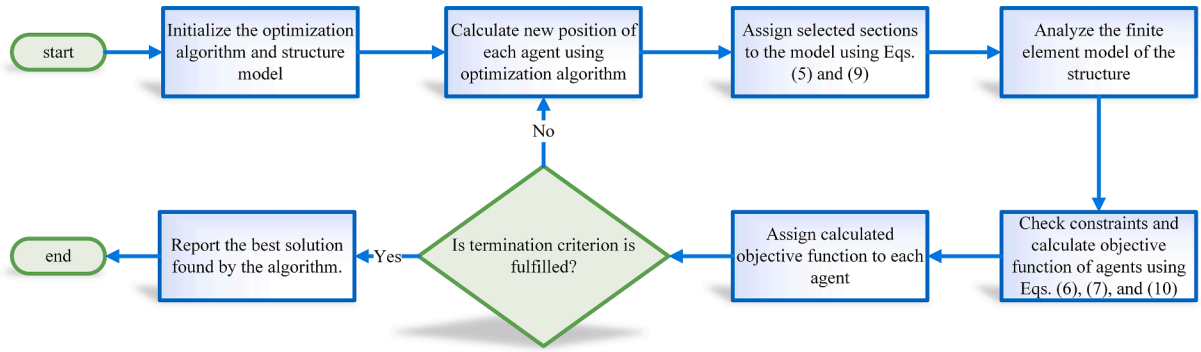

Fig. 5. Flowchart for structural optimization with discrete variables.

various optimization problems [38]. This global search population-based metaheuristic is simple and works based on the concept that the solution found for an investigated problem is moving toward the best agent while avoiding the worst one. One of the most important features of the Jaya is that it does not require any specific algorithm parameters or tuning process so that it is called a parameter-less algorithm. However, the standard Jaya algorithm suffers from some shortcomings. For instance, unwanted premature convergence and weak exploration ability are the disadvantages of the standard Jaya. Like other metaheuristics, Jaya starts with an initial solution in which each variable generated randomly in the range of lower and upper bounds. After that, each variable in each candidate solution is updated iteratively in the body of the algorithm according to the following equation:

$$x'(j,c,i) = x(j,c,i) + stepsize(j,c,i) \tag{1}$$

in which $stepsize(j,c,i)$ are obtained as follows:

$$stepsize(j,c,i) = r_1 \times (x(j,b,i) - |x(j,c,i)|) - r_2 \times (x(j,w,i) - |x(j,c,i)|) \tag{2}$$

where b and w are the indexes of the best and worst solutions among the current population. Index of i, j, and c respectively denote the current iteration ($i = 1, 2, \cdots, Maxit$), design variable ($j = 1, 2, \cdots, d$), and candidate solution ($c = 1, 2, \cdots, NP$); $x(j,c,i)$ indicates the jth variable of cth candidate solution in ith iteration; r_1 and r_2 are two random numbers generated in $[0,1]$ interval. The terms $r_1 \times (x(j,b,i) - |x(j,c,i)|)$ and $-r_2 \times (x(j,w,i) - |x(j,c,i)|)$ respectively illustrate moving toward the best solution and avoiding the worst one among the current population; $x'(j,c,i)$ as the newly generated solution is chosen to replace the current solution (i.e. $x(j,c,i)$) if the objective function value gets better (replacement strategy). This iterative process is continued until a stopping criterion is met. The schematic representation of Eq. (2) can be seen in Fig. 1.

3. Self-adaptive Multi-population based Jaya (SAMP) algorithm

The self-adaptive Multi-population based Jaya (SAMP-Jaya) algorithm has been proposed by Rao and Saroj in 2017 [23]. Maintaining the overall diversity of the search space by splitting the entire population into several groups and having the ability to search in different regions by distributing solutions over the search space simultaneously are the advantages of this optimization method in comparison to the standard Jaya algorithm. In this method, first of all, the entire population is sorted in ascending order based on the objective function values of the solutions. After that, the population is grouped into m subpopulations or groups (Initially $m = 2$ is considered) based on the quality of the solutions. To generate subpopulation 1, the first $a_1 = round(NP/m \times 1)$ members from the entire sorted population are selected and assigned to the subpopulation 1. Next, the $a_1 + 1$ through $a_2 = round(NP/m \times 2)$ members are selected from the rest of the sorted population and assigned to the subpopulation 2. The process of the population portioning will be

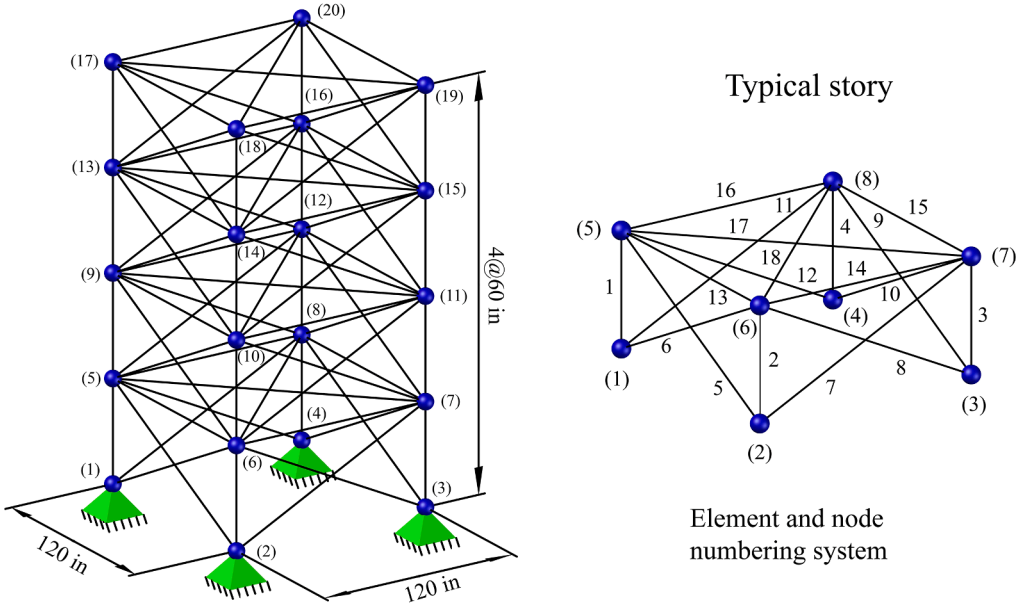

Fig. 6. The 72-bar spatial truss structure.

Table 1
The List of the cross-sectional areas from the AISC code for the 72-bar spatial truss structure.

No.	in.^{2}	No.	in.^{2}	No.	in.^{2}	No.	in.^{2}
1	0.111	17	1.563	33	3.840	49	11.500
2	0.141	18	1.620	34	3.870	50	13.500
3	0.196	19	1.80	35	3.880	51	13.900
4	0.25	20	1.990	36	4.180	52	14.200
5	0.307	21	2.130	37	4.220	53	15.500
6	0.391	22	2.380	38	4.490	54	16.000
7	0.442	23	2.620	39	4.590	55	16.900
8	0.563	24	2.630	40	4.800	56	18.800
9	0.602	25	2.880	41	4.970	57	19.900
10	0.766	26	2.930	42	5.120	58	22.000
11	0.785	27	3.090	43	5.740	59	22.900
12	0.994	28	3.130	44	7.220	60	24.500
13	1.000	29	3.380	45	7.970	61	26.500
14	1.228	30	3.470	46	8.530	62	28.000
15	1.266	31	3.550	47	9.300	63	30.000
16	1.457	32	3.630	48	10.850	64	33.500

Table 2
Load cases for the 72-bar spatial truss structure.

Case	Node	F_x (kips)	F_y (kips)	F_z (kips)
1	17	5.0	5.0	−5.0
2	17	0.0	0.0	−5.0
	18	0.0	0.0	−5.0
	19	0.0	0.0	−5.0
	20	0.0	0.0	−5.0

continued until all m subpopulations are generated. The subpopulations generation are shown schematically in Fig. 2. In each subpopulation, the following procedure is performed:

a) the best and worst solutions are determined.
b) the solutions are modified according to Eq. (1)
c) solutions are updated using the replacement strategy.

This process is executed for all m subpopulations independently in each iteration. In the following, the entire subpopulations are merged. Then, m will be increased ($m = m + 1$), If the best solution in the current iteration is better than that specified in the previous iteration. Otherwise, m will be decreased (i.e.$m = m-1$). After each iteration, the stopping criterion is checked. If the algorithm satisfies the termination condition, the best optimum solution obtained so far is reported. Otherwise, $i = i+1$ and the algorithm starts from sorting population in the next iteration. All above-mentioned process is repeated until the algorithm reaches to *Maxit* as a termination condition. The most important feature of the SAMP-Jaya algorithm is that the whole population is divided into some groups based on the quality of the solution. Furthermore, during the search process these groups are adaptively changed (decreased or increased) based on the strength of the solution change.

4. Improved Shuffled Jaya algorithm

Many optimization algorithms can be easily embedded into multi-population methods. Using multi-population is one of the most effective techniques to maintain population diversity. Maintaining population diversity by distributing candidate solutions over the entire search space is the main aim of multi-population methods. This feature considerably makes optimization methods find a global optimum solution efficiently. The number of subpopulations, communication between subpopulations, search area of subpopulations, and search mechanism of subpopulations are the important issues for the designing multi-population metaheuristic [22]. The first issue deals with determining the number of sub-populations. Using a fixed number of subpopulations and a varying number of subpopulations are two ways to indicate the first issue. Simplicity and requiring only a certain value specified based on the researchers' experience are the advantages of the fixed number of subpopulations. However, determining the number of subpopulations may difficult for different optimization problems. Exchanging information between different subpopulations as the second issue is very useful for optimization due to improving the searchability of algorithms. Moreover, it can lead to finding promising solutions over the search space. The third issue deals with determining the search area of each subpopulation over the search space. A too-small search area for each subpopulation can cause the subpopulation to converge to local minima. In contrary, a too big search area for each subpopulation may cause subpopulation works such as a single-population optimization algorithm. In this regard, it is required to consider an appropriate search area for each sub-population in the entire search space. The last issue is related to determining the search mechanism of each subpopulation. For example, Wu et al [39] proposed a DE algorithm with multi-population in which each subpopulation applied different mutation strategies. The

Table 3
Comparison results of different optimization methods for the 72-bar spatial truss.

Element group	Optimal cross-sectional areas									
	SGA [42]	CBO [43]	IMBA [44]	aeDE [35]	Accelerated WEO [45]	BBO-DE [46]	EFA [47]	Jaya	SAMP-Jaya	IS-Jaya
A1	0.196	1.620	1.990	1.990	1.99	1.990	1.990	1.990	1.990	1.990
A2	0.602	0.563	0.442	0.563	0.563	0.563	0.563	0.563	0.563	0.563
A3	0.307	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
A4	0.766	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
A5	0.391	1.457	1.228	1.228	1.228	1.228	1.228	1.228	1.228	1.228
A6	0.391	0.442	0.563	0.442	0.442	0.442	0.442	0.442	0.563	0.563
A7	0.141	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
A8	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
A9	1.800	0.602	0.563	0.563	0.563	0.563	0.563	0.563	0.563	0.563
A10	0.602	0.563	0.563	0.563	0.563	0.563	0.563	0.563	0.442	0.442
A11	0.141	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
A12	0.307	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
A13	1.563	0.196	0.196	0.196	0.196	0.196	0.196	0.196	0.196	0.196
A14	0.766	0.602	0.563	0.563	0.563	0.563	0.563	0.563	0.563	0.563
A15	0.141	0.391	0.391	0.391	0.391	0.391	0.391	0.391	0.391	0.391
A16	0.111	0.563	0.563	0.563	0.563	0.563	0.563	0.563	0.563	0.563
Best weight (lb)	427.203	391.07	389.334	389.334	389.33	389.33	389.334	389.3342	389.3342	389.3342
NSAs	60,000	4500	6250	4160	7490	5840	2700	3740	5980	2680
Worst weight (lb)	N/A	495.97	389.823	393.325	397.74	394.98	393.826	417.9578	429.3339	392.3749
Mean weight (lb)	N/A	403.71	389.457	390.913	390.94	390.62	391.376	395.1115	398.0957	389.9360
Standard deviation (lb)	N/A	24.8	0.84	1.161	1.93	1.43	1.376	11.2985	14.6093	0.8202

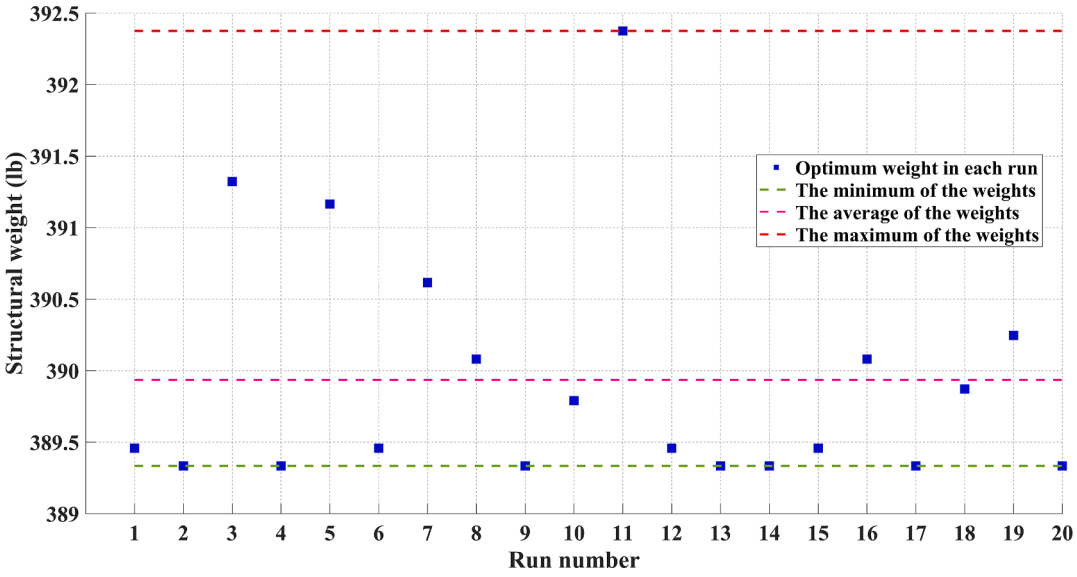

Fig. 7. The obtained structural weight in each independent run for the 72-bar spatial truss structure.

search mechanism has a considerably positive influence on the performance of multi-population methods. This mechanism is also a promising way to enhance the performance of the optimization algorithms.

Different population partitioning into subpopulations techniques under both fixed and varying number of subpopulations have been recently studied in Ref. [40]. Both Shuffled Complex Evolution algorithm (SCE) [4] and Swarm Bat Algorithm with Improved Search (SBAIS) [41] have been developed with different population partitioning techniques based on shuffling. In these methods, the shuffling process denotes to merging subpopulations into one population and generating new subpopulations. SCE uses the quality of candidate solutions to dividing them into several subpopulations called complexes, each of which evolves independently. After evolution, the complexes are merged, sorted based on the fitness function, and divided into complexes again. In SCE, after sorting candidate solutions based on their fitness value, the population is partitioned into m complexes $(m_1, m_2, \cdots, m_m)$. Next, the best m solutions $(x_1, x_2, \cdots, x_m)$ are selected and assigned to complexes $m_1, m_2, \cdots, m_m$, respectively. In the following, the next best solutions $(x_{m+1}, x_{m+2}, \cdots, x_{2m})$ are respectively assigned to the complexes $(m_1, m_2, \cdots, m_m)$. This process repeats for other solutions until all of them are assigned to the complexes. The difference between SCE and SBAIS is that SBAIS starts with assigning the first NP/m solutions per swarm to the first swarm (called master subpopulation), while SCE starts with assigning respectively the first m solutions to the m complexes. The continuing process of population portioning in SBAIS is according to the partitioning technique presented in SCE.

By considering all of the above-mentioned significant points, here, a new variant of Jaya algorithm namely Improved Shuffled based Jaya (IS-Jaya) algorithm is proposed. IS-Jaya algorithm works based on multi-population methods. Once initialized, the objective function value of all NP agents in the population are computed and sorted based on these values. To divide NP agents into fixed m subpopulations numbers (communities), IS-Jaya selects the first m solutions to be placed randomly into the m communities. Next, the $m+1, m+2, \cdots, 2m$ candidate solutions are selected from the rest of the population and again randomly assigned to the m communities. This process continues until all the NP agents are assigned to the m communities. Fig. 3 shows how to partition NP agents into m communities. As can be seen, there is a

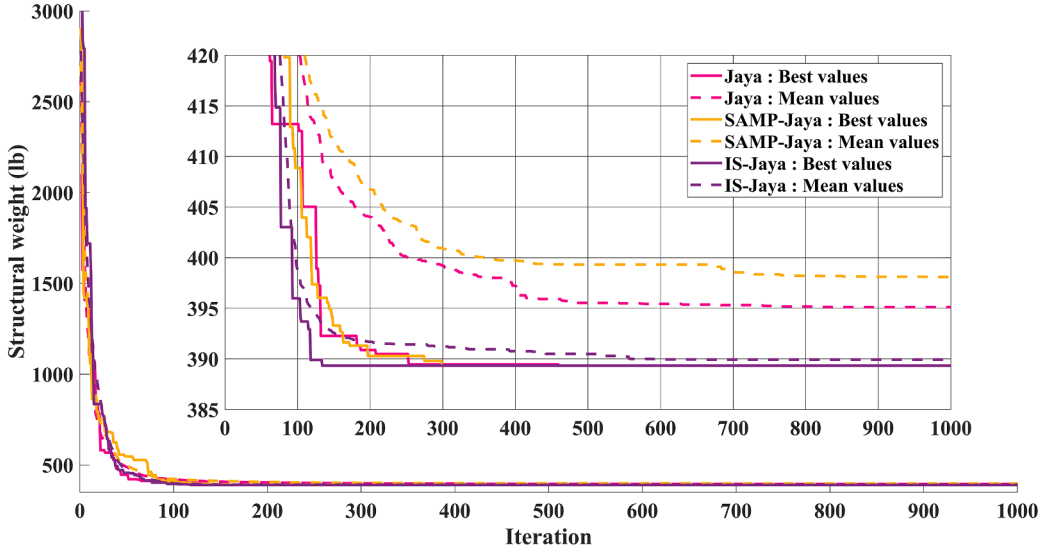

Fig. 8. Convergence history of the Jaya, SAMP-Jaya, and IS-Jaya for the 72-bar spatial truss structure.

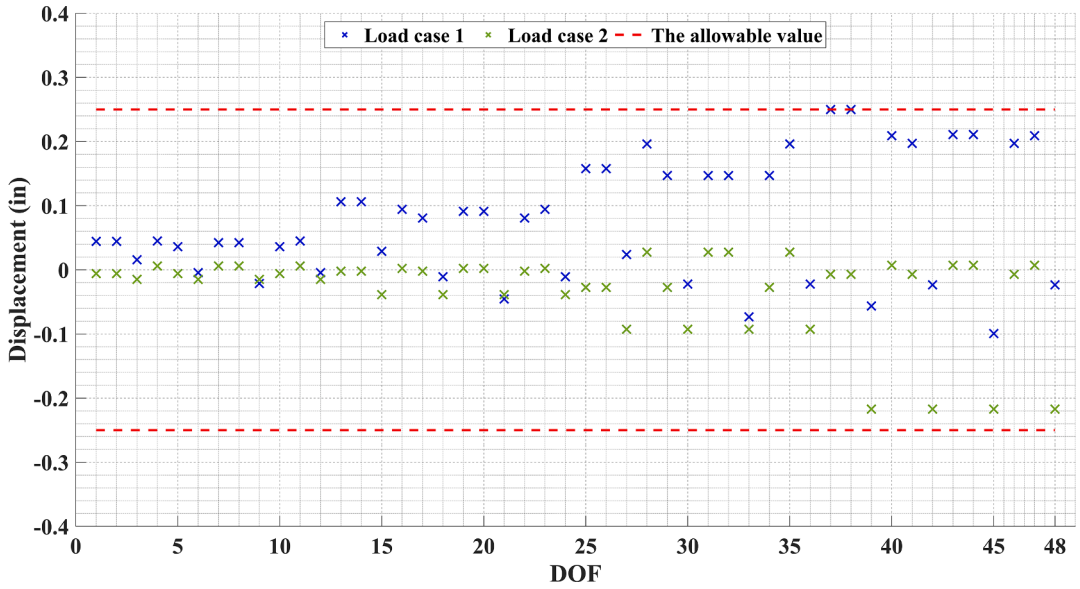

Fig. 9. Displacement values calculated at the obtained optimal design by the IS-Jaya for the 72-bar spatial truss structure.

substantial difference between the population portioning technique in SCE and IS-Jaya algorithm. In SCE, the m candidate solutions are respectively assigned to m complexes each time. However, in IS-Jaya, they are randomly assigned to the m communities. This random assignment increases population diversity in comparison to SCE. After the building communities in the IS-Jaya algorithm, each of these communities is permitted to undergoes independent evolution based on the standard Jaya algorithm. Dividing the entire population into several communities by this strategy considerably helps the Jaya algorithm to search different areas of the search space simultaneously. After evolution at each iteration, the communities are merged and their information are shared through the process of shuffling. Shuffling the communities helps to share information, independently acquired by each community. Moreover, it makes to avoid unwanted premature convergence and maintain population diversity. To avoid falling into the local optimal and improve the performance of the original Jaya algorithm, a mechanism is utilized to escape from local optima in each community. This mechanism provides opportunities for the members in each community to move all over the search space so that it improves the diversification of the search space.

In a general view, the proposed IS-Jaya algorithm uses the shuffling process to share information among communities while the standard Jaya algorithm does not have this process. Using this process leads to the exploration ability of the IS-Jaya algorithm becomes more than the standard Jaya algorithm. Moreover, the IS-Jaya algorithm has a mechanism to escape from local optima, whereas the standard Jaya suffers from this shortcoming. On the other hand, the substantial differences between SAMP-Jaya algorithm and IS-Jaya algorithm are summarizing as follows:

A. The main difference deals with how to partition the entire population into several subpopulations. In SAMP-Jaya, the members from $a_{i-1}+1$ to a_i are selected from the entire sorted population and assigned to subpopulation i. Nevertheless, in the IS-Jaya algorithm, the m members of the entire sorted population are respectively selected and randomly assigned to communities. For further clarity, Fig. 2

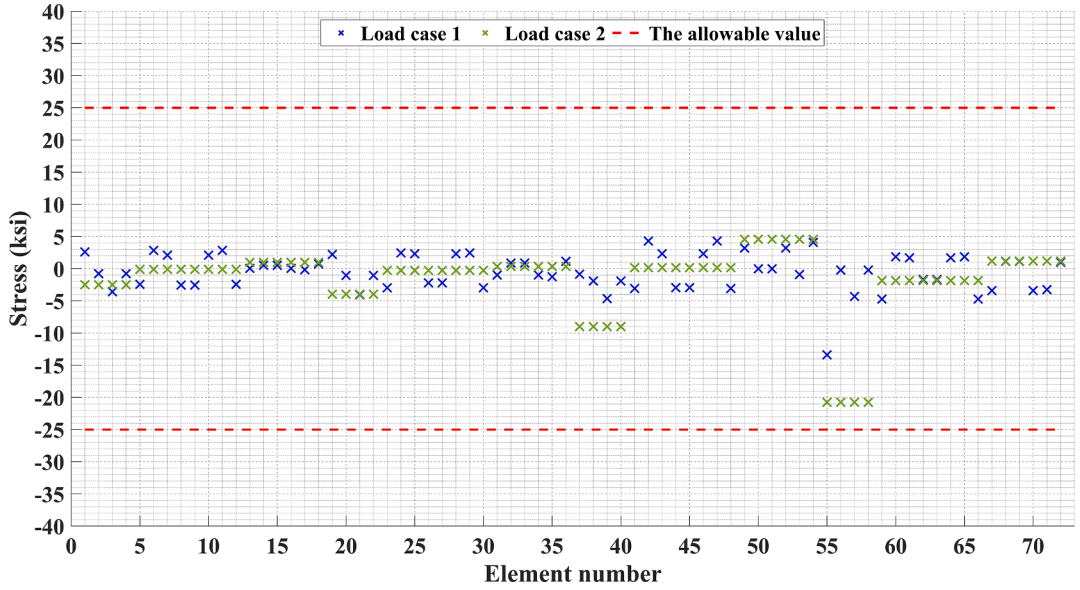

Fig. 10. Stress values calculated at the obtained optimal design by the IS-Jaya for the 72-bar spatial truss structure.

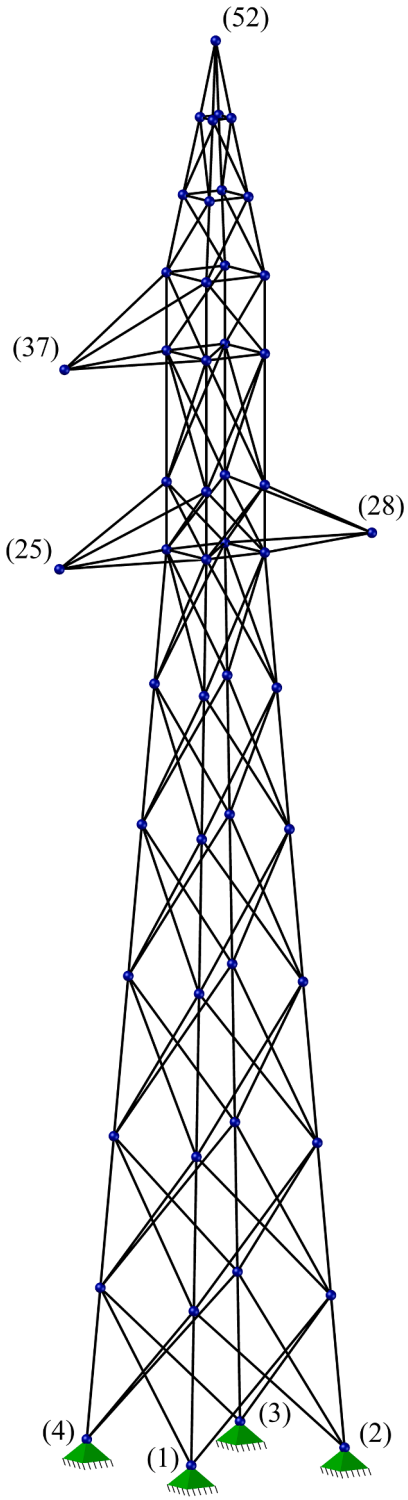

Fig. 11. The 160-bar spatial truss structure.

and Fig. 3 show the main difference between SAMP-Jaya and IS-Jaya. In IS-Jaya, population partitioning is in a way that makes the exploration around the available best solutions. Nevertheless, in SAMP-Jaya, this partitioning is in a way that leads the best solution assigns to the first subpopulation and cannot search around the available best solutions. This results that IS-Jaya has better exploration ability than SAMP-Jaya.

B. The second difference is related to a fixed and varying number of subpopulations. In SAMP, the size of subpopulations continuously changes during the search space, while this value is constant in the IS-Jaya algorithm.

C. IS-Jaya algorithm uses a mechanism to escape from local minima, whereas SAMP does not have this useful mechanism. Adding this mechanism to IS-Jaya makes the algorithm find a better optimum solution than SAMP-Jaya. Moreover, the probability of falling into local optima is decreased in comparison to SAMP-Jaya.

4.1. Steps of IS-Jaya algorithm

The flowchart of the proposed IS-Jaya algorithm is illustrated in Fig. 4 and the steps of the algorithm involved given in the following:

Step 1: Initialization

Like other metaheuristic algorithms, IS-Jaya algorithm starts with initial positions for all agents generated randomly in a d-dimensional search space as:

$$x_{j,c,0} = x_{min} + r \times (x_{max} - x_{min}); c = 1, 2, \cdots, NP \tag{3}$$

where x_{min} and x_{max} are the lower and upper bounds of the jth design variable for the cth agent. r is a random number uniformly distributed in the range of $[0, 1]$.

Step 2: Partitioning into communities

In this step, the entire population with the size of NP is partitioned into m communities. For this purpose, the entire population is sorted in ascending order based on the objective function value of the solutions. Next, to partition the population into communities, the first m candidate solutions (members) of the entire population (i.e. NP) are selected and randomly placed in each community so that each community has one member so far. After that, the second m members are selected from the rest of the NP and again placed randomly as the second members of each community. This process continues until all the solutions are assigned to the communities.

Step 3: Generating a new solution

In each community, the best and worst solutions are determined and the new solution is then generated using Eq. (1).

Step 4: Escaping from local optima

Metaheuristics are trapped in local optima when they are close to the optimum solution. In order to tackle this issue, one dimension of a randomly selected member in each community after generating its new position according to step 3 is randomly selected and changed by the following equation:

$$x'_{q,r,i} = x'_{q,r,i} + 0.1 \times rn \times (x_{max} - x_{min}) \tag{4}$$

in which q and r denote respectively the index of the selected variable for the selected agent. rn is normally distributed random number.

Step 5: Evaluating and applying replacement strategy

In this step, the newly generated solutions are evaluated in each community. The new solution will be replaced by the existing solution if the new solution is better than it. Otherwise, the existing solution will be kept unchanged.

Step 6: Merge communities for shuffling

In order to share information among communities, they are merged. Therefore, a single population is formed. Shuffling the population and forming new communities make sure that the information independently acquired in each community is shared with other communities. This process improves the exploration capability of the IS-Jaya algorithm as well as its exploitation.

Step 7: checking termination condition

If the iteration number reaches the maximum number of iterations (Maxit), the algorithm terminates. Otherwise, it returns to Step 2 for the next round of iteration.

Table 4
The nodal coordinate of the 160-bar spatial truss structure.

Node no.	X (cm)	Y(cm)	Z (cm)	Node no.	X (cm)	Y(cm)	Z (cm)
1	−105	−105	0	27	40	−40	1027.5
2	105	−105	0	28	214	0	1027.5
3	105	105	0	29	40	40	1027.5
4	−105	105	0	30	−40	40	1027.5
5	−93.929	−93.929	175	31	−40	−40	1105.5
6	93.929	−93.929	175	32	40	−40	1105.5
7	93.929	93.929	175	33	40	40	1105.5
8	−93.929	93.929	175	34	−40	40	1105.5
9	−82.859	−82.859	350	35	−40	−40	1256.5
10	82.859	−82.859	350	36	40	−40	1256.5
11	82.859	82.859	350	37	−207	0	1256.5
12	−82.859	82.859	350	38	40	40	1256.5
13	−71.156	−71.156	535	39	−40	40	1256.5
14	71.156	−71.156	535	40	−40	−40	1346.5
15	71.156	71.156	535	41	40	−40	1346.5
16	−71.156	71.156	535	42	40	40	1346.5
17	−60.085	−60.085	710	43	−40	40	1346.5
18	60.085	−60.085	710	44	−26.592	−26.592	1436.5
19	60.085	60.085	710	45	26.592	−26.592	1436.5
20	−60.085	60.085	710	46	26.592	26.592	1436.5
21	−49.805	−49.805	872.5	47	−26.592	26.592	1436.5
22	49.805	−49.805	872.5	48	−12.737	−12.737	1526.5
23	49.805	49.805	872.5	49	12.737	−12.737	1526.5
24	−49.805	49.805	872.5	50	12.737	12.737	1526.5
25	−214	0	1027.5	51	−12.737	12.737	1526.5
26	−40	−40	1027.5	52	0	0	1615

Table 5
End nodes of the members of the 160-bar spatial truss structure.

Elem. no.	Node		A_i	Elem. no.	Node		A_i	Elem. no.	Node		A_i	Elem. no.	Node		A_i
	1	2			1	2			1	2			1	2	
1	1	5	1	41	13	18	8	81	25	31	17	121	36	40	29
2	2	6	1	42	14	17	8	82	28	32	17	122	38	41	29
3	3	7	1	43	14	19	8	83	28	33	17	123	39	42	29
4	4	8	1	44	15	18	8	84	25	34	17	124	35	43	29
5	1	6	2	45	15	20	8	85	26	31	18	125	40	41	30
6	2	5	2	46	16	19	8	86	27	32	18	126	41	42	30
7	2	7	2	47	16	17	8	87	29	33	18	127	42	43	30
8	3	6	2	48	13	20	8	88	30	34	18	128	43	40	30
9	3	8	2	49	17	21	9	89	26	32	19	129	35	36	31
10	4	7	2	50	18	22	9	90	27	31	19	130	36	38	31
11	4	5	2	51	19	23	9	91	29	34	19	131	38	39	31
12	1	8	2	52	20	24	9	92	30	33	19	132	39	35	31
13	5	9	3	53	17	22	10	93	27	33	20	133	40	44	32
14	6	10	3	54	18	21	10	94	29	32	20	134	41	45	32
15	7	11	3	55	19	24	10	95	30	31	20	135	42	46	32
16	8	12	3	56	20	23	10	96	26	34	20	136	43	47	32
17	5	10	4	57	18	23	11	97	26	29	21	137	40	45	33
18	6	9	4	58	19	22	11	98	27	30	21	138	41	46	33
19	6	11	4	59	20	21	11	99	31	35	22	139	42	47	33
20	7	10	4	60	17	24	11	100	32	36	22	140	43	44	33
21	7	12	4	61	21	26	12	101	33	38	22	141	44	45	34
22	8	11	4	62	22	27	12	102	34	39	22	142	45	46	34
23	8	9	4	63	23	29	12	103	33	39	23	143	46	47	34
24	5	12	4	64	24	30	12	104	32	35	23	144	44	47	34
25	9	13	5	65	21	27	13	105	31	36	23	145	44	48	35
26	10	14	5	66	22	26	13	106	34	38	23	146	45	49	35
27	11	15	5	67	23	30	13	107	32	38	24	147	46	50	35
28	12	16	5	68	24	29	13	108	33	36	24	148	47	51	35
29	9	14	6	69	22	29	14	109	34	35	24	149	45	48	36
30	10	13	6	70	23	27	14	110	31	39	24	150	46	49	36
31	10	15	6	71	24	26	14	111	37	35	25	151	47	50	36
32	11	14	6	72	21	30	14	112	37	39	25	152	44	51	36
33	11	16	6	73	26	27	15	113	37	40	26	153	48	49	37
34	12	15	6	74	27	29	15	114	37	43	26	154	49	50	37
35	12	13	6	75	29	30	15	115	35	40	27	155	50	51	37
36	9	16	6	76	30	26	15	116	36	41	27	156	48	51	37
37	13	17	7	77	25	26	16	117	38	42	27	157	48	52	38
38	14	18	7	78	27	28	16	118	39	43	27	158	49	52	38
39	15	19	7	79	25	30	16	119	35	38	28	159	50	52	38
40	16	20	7	80	29	28	16	120	36	39	28	160	51	52	38

Table 6
The List of the cross-sectional areas used for the 160-bar spatial truss structure.

No.	A (cm^2)	R (cm)	No.	A (cm^2)	R (cm)	No.	A (cm^2)	R (cm)
1	1.84	0.47	15	9.40	1.35	29	33.90	2.33
2	2.26	0.57	16	10.47	1.36	30	34.77	2.97
3	2.66	0.67	17	11.38	1.45	31	39.16	2.54
4	3.07	0.77	18	12.21	1.55	32	43.00	2.93
5	3.47	0.87	19	13.79	1.75	33	45.65	2.94
6	3.88	0.97	20	15.39	1.95	34	46.94	2.94
7	4.79	0.97	21	17.03	1.74	35	51.00	2.92
8	5.27	1.06	22	19.03	1.94	36	52.10	3.54
9	5.75	1.16	23	21.12	2.16	37	61.82	3.96
10	6.25	1.26	24	23.20	2.36	38	61.90	3.52
11	6.84	1.15	25	25.12	2.57	39	68.30	3.51
12	7.44	1.26	26	27.50	2.35	40	76.38	3.93
13	8.06	1.36	27	29.88	2.56	41	90.60	3.92
14	8.66	1.46	28	32.76	2.14	42	94.13	3.92

5. Structural optimization with discrete design variables

Finding the optimum weight of skeletal structures (trusses and frames) with discrete design variables is known as a nonlinear programming problem. It is also an efficient way to evaluate the new optimization methods when these structural optimization problems include a large number of design variables and multi-nonlinear constraints. In order to minimize the cross-sectional area of structural elements, they are divided into several element groups due to structural symmetry. Then, the number of element groups is considered equal to the number of design variables so that each element group has the same cross-sectional area. In other words, the cross-sectional area of each element group, which its elements have the same cross-section, is a discrete design variable. All cross-sectional areas are selected from a list of discrete cross-sections. These structural optimization problems aim to find the minimum-weight design while satisfying design constraints on element stresses and nodal displacements. Thus, the optimization problem definition for the discrete size optimization of the skeletal structures can be stated mathematically as follows:

$$\text{Find}\{X\} = \{x_1, x_2, \cdots, x_{neg}\}; x_i \in D_i \tag{5}$$

$$\text{Tominimize} : W(\{X\}) = \sum_{i=1}^{ne} \rho_i \cdot x_i \cdot L_i \tag{6}$$

$$\text{Subjectto} : dc_j(\{X\}) \le 0; j = 1, 2, \cdots, nc \tag{7}$$

where $\{X\}$ is the set of design variables; *neg* is the number of element groups (number of design variables); D_i is the allowable set of values for the design variable x_i; $W(\{X\})$ is the weight of the whole skeletal structure; *ne* is the total number of members in the skeletal structure; ρ_i, x_i, and L_i represent respectively material density, cross-sectional area, and length of the *i*th member; $dc_j(\{X\})$ is the *i*th design constraint of the optimization problem; and *nc* is the number of constraints in the problem.

Design variables can be chosen either from a discrete set or from a continuous one. In the case of a continuous optimization problem, design variables can vary continuously between the lower and upper bounds of the search space as:

$$D_i = \{x_i | x_i \in [x_{i,min}, x_{i,max}]\} \tag{8}$$

in which $x_{i,min}$ and $x_{i,max}$ are the allowable lower and upper bounds for the design variable x_i, respectively. For a discrete optimization problem, the design variables represent a selection from a set of parts as follows:

$$D_i = \{d_{i,1}, d_{i,2}, \cdots, d_{i,r(i)}\} \tag{9}$$

where $r(i)$ is the number of available discrete values for the *i*th design variable.

To handle the constraints of the optimization problem, the well-known penalty function method is employed as follows:

$$f_{penalty}(\{X\}) = (1 + \varepsilon_1 \cdot \nu)^{\varepsilon_2} \times W(\{X\}); \nu = \sum_{i=1}^{nc} max\{0, dc_j(\{X\})\} \tag{10}$$

where ν denotes the sum of the design violated constraints; ε_1 and ε_2 are fixed selected to make a good balance between the exploration and exploitation rate of the search space. In this paper, ε_1 is set to 1, while ε_2 starts at 1.5 and then increases linearly to 3 at the last step of the search process. For further clarity, Fig. 5 indicates how the structural optimization with discrete variables is worked, that facilitates the presentation of this optimization problem.

This paper examines two types of skeletal structures including three trusses and a planar steel frame structure. The constraint conditions of these structures are briefly described in Section 6.

6. Design examples

In this section, four benchmark examples are investigated to evaluate the efficiency and capability of the proposed method. These examples are including a 72-bar spatial truss, a 160-bar spatial, a large scale 942-bar tower truss, and a 3-bay 24-story steel plain frame structure of a building. In all test examples, the results obtained by the IS-Jaya algorithm are compared with those results found by Jaya and SAMP-Jaya algorithms. Moreover, they are compared with other existing optimization methods. The entire population in all design examples are considered equal to 20 (NP = 20). The number of communities is

Table 7
Loading conditions for the 160-bar spatial truss structure.

Lod case	Node	P_x	P_y	P_z	Lod case	Node	P_x	P_y	P_z
1	52	−868	0	−491	5	52	−917	0	−491
	37	−996	0	−546		37	−951	0	−546
	25	−1091	0	−546		25	−1015	0	−546
	28	−1091	0	−546		28	−636	1259	−428
2	52	−493	1245	−363	6	52	−917	0	−491
	37	−996	0	−546		37	−572	1303	−428
	25	−1091	0	−546		25	−1015	0	−546
	28	−1091	0	−546		28	−1015	0	−546
3	52	−917	0	−491	7	52	−917	0	−491
	37	−951	0	−546		37	−951	0	−546
	25	−1015	0	−546		25	−1015	0	−546
	28	−1015	0	−546		28	−636	1303	−428
4	52	−917	0	−546	8	52	−498	1460	−363
	37	−572	1259	−428		37	−951	0	−546
	25	−1015	0	−546		25	−1015	0	−546
	28	−1015	0	−546		28	−1015	0	−546

Table 8
Comparison results of different optimization methods for the 160-bar spatial truss structure.

Element group	Optimal cross-sectional areas (cm^2)						
	RGA [49]	RBAS [50]	aeDE [35]	EFA [47]	Jaya	SAMP-Jaya	IS-Jaya
A1	19.03	19.03	19.03	19.03	19.03	19.03	19.03
A2	5.27	5.27	5.27	5.27	5.27	5.27	5.27
A3	19.03	19.03	19.03	19.03	19.03	19.03	19.03
A4	5.27	5.27	5.27	5.27	5.27	5.27	5.27
A5	19.03	19.03	19.03	19.03	19.03	19.03	19.03
A6	5.75	5.75	5.75	5.75	5.75	5.75	5.75
A7	15.39	15.39	15.39	15.39	15.39	15.39	15.39
A8	5.75	5.75	5.75	5.75	5.75	5.75	5.75
A9	13.79	13.79	13.79	13.79	13.79	13.79	13.79
A10	5.75	5.75	5.75	5.75	5.75	5.75	5.75
A11	5.75	5.75	5.75	5.75	5.75	5.75	5.75
A12	13.79	12.21	12.21	12.21	12.21	12.21	12.21
A13	6.25	6.25	6.25	6.25	6.25	6.25	6.25
A14	5.75	5.75	5.75	5.75	5.75	5.75	5.75
A15	2.66	3.47	3.88	3.88	3.88	3.47	3.88
A16	7.44	7.44	7.44	7.44	7.44	7.44	7.44
A17	1.84	1.84	1.84	1.84	1.84	1.84	1.84
A18	8.66	9.4	8.66	8.66	8.66	8.66	8.66
A19	2.66	2.66	2.66	2.66	2.66	2.66	2.66
A20	3.07	3.47	3.07	3.07	3.07	3.07	3.07
A21	2.66	3.07	2.66	2.66	2.66	3.47	2.66
A22	8.06	8.06	8.06	8.06	8.06	8.06	8.06
A23	5.27	5.27	5.75	5.75	5.75	5.75	5.75
A24	6.25	6.25	6.25	6.25	6.25	6.25	6.25
A25	5.75	5.75	5.75	6.25	6.25	5.75	5.75
A26	1.84	2.26	2.26	1.84	1.84	2.26	2.26
A27	4.79	4.79	4.79	4.79	4.79	4.79	4.79
A28	2.66	3.07	2.66	2.66	2.66	2.66	2.66
A29	3.47	3.47	3.47	3.47	3.47	3.47	3.47
A30	1.84	1.84	1.84	1.84	1.84	1.84	1.84
A31	2.26	3.88	2.26	2.26	2.26	2.26	2.26
A32	3.88	3.88	3.88	3.88	3.88	3.88	3.88
A33	1.84	1.84	1.84	1.84	1.84	1.84	1.84
A34	1.84	2.26	1.84	1.84	1.84	1.84	1.84
A35	3.88	3.88	3.88	3.88	3.88	3.88	3.88
A36	1.84	2.26	1.84	1.84	1.84	1.84	1.84
A37	1.84	3.47	1.84	1.84	1.84	1.84	1.84
A38	3.88	3.88	3.88	3.88	3.88	3.88	3.88
Best weight (kg)	1337.442	1348.905	1336.634	1336.704	1336.704	1337.043	1336.634
NSAs	N/A	90,000	23,925	16,870	18,160	17,780	11,740
Worst weight (kg)	N/A	1401.6323	1410.611	1429.253	1399.572	1420.340	1366.933
Mean weight (kg)	N/A	1367.5275	1355.875	1372.551	1356.540	1355.328	1342.807
Standard deviation (kg)	N/A	N/A	18.805	34.706	17.528	20.691	8.649

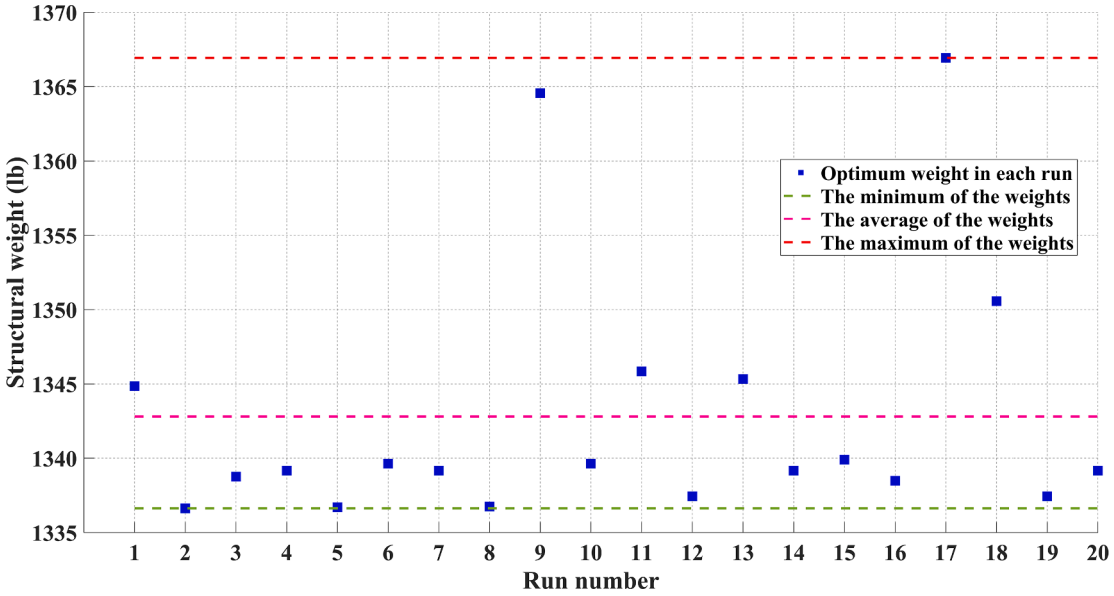

Fig. 12. The obtained structural weight in each independent run for the 160-bar spatial truss structure.

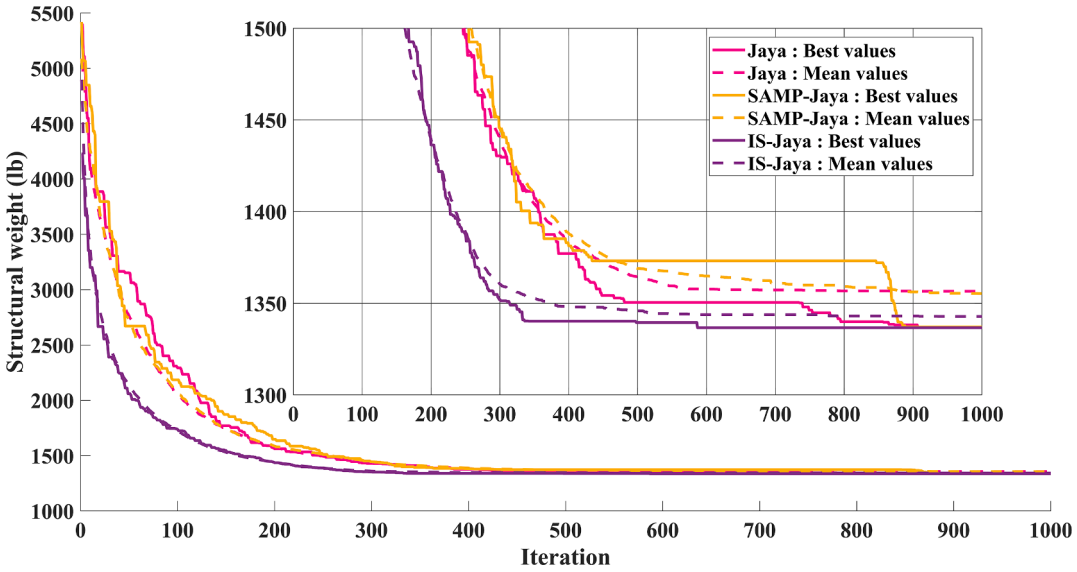

Fig. 13. Convergence history of the Jaya, SAMP-Jaya, and IS-Jaya for the 160-bar spatial truss structure.

considered equal to m = 4 for the IS-Jaya algorithm, while m is adaptively changed in SAMP-Jaya as mentioned in Section 3. The maximum number of structural analyses in all test examples is set to be 20,000 except the third example that is equal to 60000. 20 independent runs are performed to get statistically meaningful results.

6.1. A 72-bar spatial truss structure

The first design example is considered a 72-bar spatial truss structure as shown in Fig. 6. This benchmark truss has been optimized by many researchers [35,42–47]. The material density and the modulus of elasticity of the truss are respectively $0.1lb/in^3$ and 10^4ksi. The 72 elements of the truss are grouped into 16 element groups due to structural symmetry: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}–A_{22}, (6) A_{23}–A_{30}, (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66}, (15) A_{67}–A_{70}, and (16) A_{71}–A_{72}. The obtained stress in all elements must not exceed from $\pm 25ksi$. In addition, all nodal displacements must not be larger than $\pm 0.25in$. The design variables are selected from 64 discrete values from 0.111 to 33.5 in^2, as given in Table 1. Two separate load cases acting on the structure are listed in Table 2.

Table 3 provides the comparison results of Jaya, SAMP-Jaya, and IS-Jaya algorithms with other available results acquired by Steady-state genetic algorithm (SGA) [42], Colliding Bodies Optimization (CBO) [43], Improved Mine Blast Algorithm (IMBA) [44], Adaptive Elitist Differential Evolution (aeDE) [35], Accelerated Water Evaporation Optimization (Accelerated-WEO) [45], hybrid of biogeography-based optimization and differential evolution methods (BBO-DE) [46], and Electromagnetism-like Firefly Algorithm (EFA) [47]. From this table, it can be seen that the average of optimal weight found by IS-Jaya (389.9360 lb) is much better than CBO (403.71 lb), aeDE (390.913 lb), Accelerated-WEO (390.94 lb), BBO-DE (390.62 lb), EFA (391.376 lb), Jaya (395.1115 lb), SAMP-Jaya (398.0957 lb) and only slightly inferior to the results found by IMBA (389.457 lb). Furthermore, a close examination of this table reveals that the number of required analyses (NSAs) found by IS-Jaya is 2680, which is smaller than those obtained by the other existing methods. In other words, although in this problem IS-Jaya like other optimization methods including IMBA, aeDE,

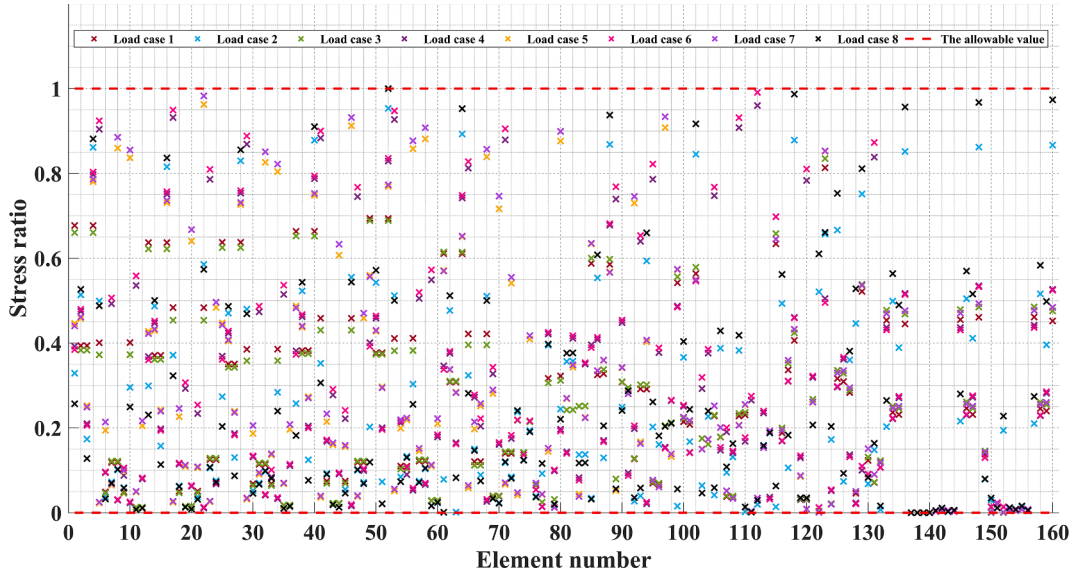

Fig. 14. Stress ratio values calculated at the obtained optimal design by the IS-Jaya for the 160-bar spatial truss structure.

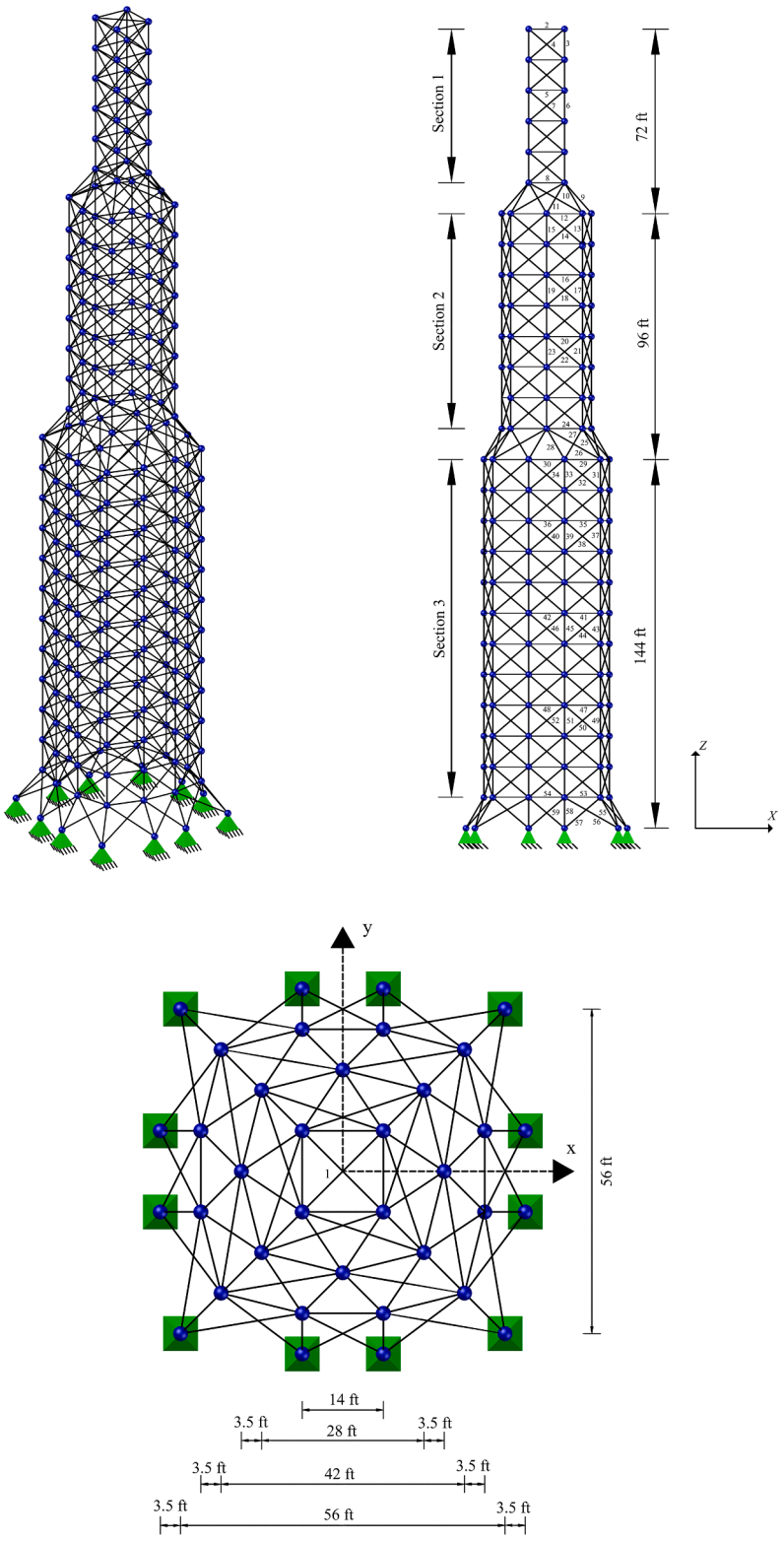

Fig. 15. The 942-bar spatial truss structure.

Accelerated WEO, BBO-DE, EFA, Jaya, and SAMP-Jaya can find the optimum solution which is equal to 389.33 lb, the NSAS obtained by IS-Jaya (2680) has the lowest value. This indicates that IS-Jaya has better performance than the other existing methods in terms of convergence speed.

Fig. 7 shows the optimum weight found by the IS-Jaya algorithm in each independent run. As can be seen, the proposed method can find the best weight (389.3342 lb) in 7 runs. The convergence histories of the Jaya, SAMP-Jaya, and IS-Jaya are indicated in Fig. 8. From this figure, it can be concluded that IS-Jaya is superior over the Jaya and SAMP-Jaya in terms of convergence speed in both best and average runs. Displacement and stress values at the obtained optimal design by the IS-Jaya algorithm are given in Figs. 9 and 10, respectively. These figures reveal that the constraints of the structure found by the algorithm have

Table 9
Comparison results of different optimization methods for the 942-bar spatial truss structure.

Element group	Optimal cross-sectional areas (in.^{2})					
	SA [52]	FA [53]	CGFA [53]	Jaya	SAMP-Jaya	IS-Jaya
A1	1	1	1	1	1	1
A2	1	1	1	1	1	1
A3	3	3	1	3	3	4
A4	1	12	1	2	1	2
A5	1	1	1	1	1	1
A6	17	15	14	16	15	15
A7	3	3	4	3	3	3
A8	7	7	5	8	4	6
A9	20	19	5	7	7	6
A10	1	2	22	17	17	28
A11	8	7	1	1	3	5
A12	7	7	4	6	7	7
A13	19	15	19	16	15	16
A14	2	2	2	2	2	2
A15	5	7	4	5	5	5
A16	1	1	1	1	2	1
A17	22	24	21	22	22	22
A18	3	3	3	3	3	3
A19	9	8	14	9	9	9
A20	1	1	1	1	1	1
A21	34	29	35	28	29	29
A22	3	4	3	4	4	5
A23	19	17	18	15	16	18
A24	27	25	24	29	23	26
A25	42	38	36	40	39	40
A26	1	4	1	3	12	3
A27	12	13	11	11	5	13
A28	16	18	14	16	11	15
A29	19	15	14	15	14	16
A30	14	15	23	18	16	17
A31	42	39	38	36	37	38
A32	4	4	3	4	3	3
A33	4	3	2	2	3	4
A34	4	4	3	3	3	3
A35	1	2	1	1	1	1
A36	1	2	1	1	1	1
A37	62	60	70	55	59	62
A38	3	4	3	3	4	3
A39	2	2	2	2	2	2
A40	4	5	3	3	3	3
A41	1	2	1	1	1	1
A42	2	4	1	3	1	8
A43	77	80	91	79	83	69
A44	3	4	3	3	4	5
A45	2	2	2	2	2	1
A46	3	5	2	3	3	5
A47	2	2	1	1	1	1
A48	3	3	1	1	1	1
A49	100	97	102	100	96	76
A50	4	4	4	3	3	3
A51	1	1	1	1	2	7
A52	4	5	3	4	4	5
A53	6	10	10	11	9	18
A54	3	3	11	3	5	16
A55	49	46	46	42	44	57
A56	1	1	1	1	1	1
A57	62	63	65	68	62	51
A58	1	3	3	5	3	7
A59	3	1	1	1	3	4
Best weight (lb)	143,436	158,743	141,860	139028.798	139744.115	138688.913
NSAs	39,834	45,000	32,500	55,700	56,920	53,420
Worst weight (lb)	N/A	167,331	147,325	276067.074	236898.014	150721.603
Mean weight (lb)	N/A	161,551	144,231	182562.693	170278.653	142903.207
Standard deviation (lb)	N/A	6432	3342	42037.068	28367.147	3171.403

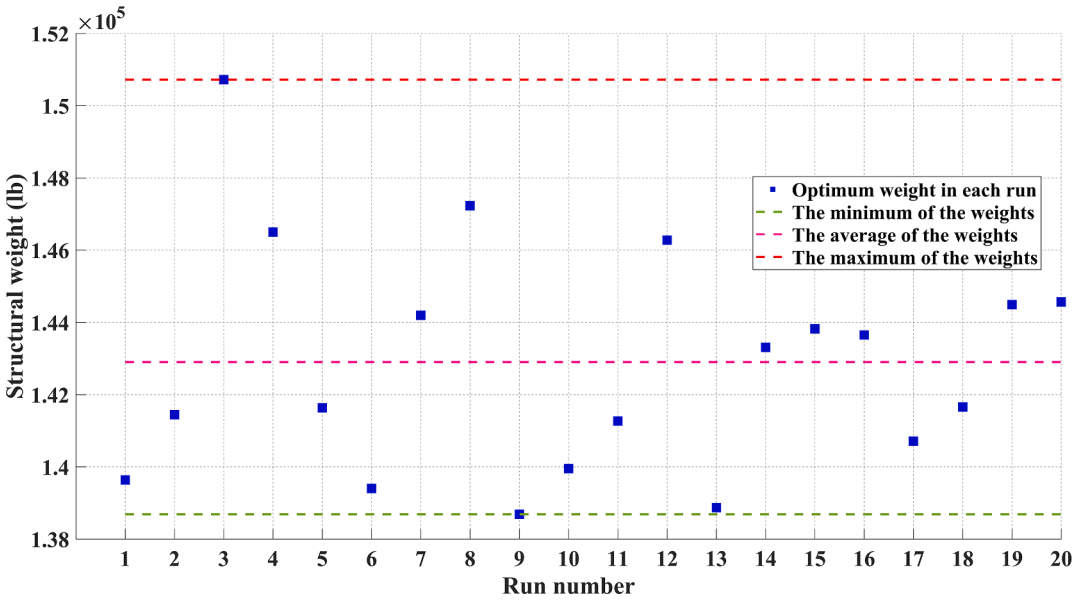

Fig. 16. The obtained structural weight in each independent run for the 942-bar spatial truss structure.

not been violated.

6.2. A 160-bar transmission tower structure

The 160-bar spatial truss structure displayed in Fig. 11 is examined as the second design example. The material density and Young's modulus are respectively $\rho = 0.00785 kg/cm^2$ and $E = 2.047 \times 10^6 kgf/cm^2$. Due to structural symmetry, 160 elements are classified into 38 element groups. Nodal coordinates and end nodes of the members are listed in Tables 4 and 5, respectively. The cross-sectional areas together with their corresponding radius of gyrations are listed in Table 6. Eight load cases acting on the truss are considered as provided in Table 7.

Obtained stress for all members (tension or compression members) must be smaller than 1500 kg/cm^2. Furthermore, for members under the compressive stress, the buckling stress limitation should be also checked as follows:

$$\sigma_i^- = \begin{cases} 1300 - \dfrac{(\lambda_i)^2}{24} if \lambda_i \le 120 \\ \dfrac{10^7}{(\lambda_i)^2} otherwise \end{cases} \tag{11}$$

where σ_i^- is the allowable compressive stress of member i; λ_i is the slender ratio for member i ($\lambda_i = kL_i/r_i$); L_i is the length of member i; r_i is the corresponding radius of gyration for member i, and k is the member effective length factor fixed as 1 for all members [48].

Table 8 summarizes the comparison results acquired by the IS-Jaya algorithm and the other optimization methods. As can be seen, while IS-Jaya and aeDE [35] give the best optimum weight (1336.634 kg), the other existing methods including RGA [49] (1337.442 kg), RBAS [50] (1348.905 kg), EFA [47] (1336.704 kg), Jaya (1336.704 kg), and SAMP-Jaya (1337.043 kg) gain larger weight. Moreover, the IS-Jaya algorithm requires 11,740 analyses to converge the optimum weight, whereas RBAS, aeDE, EFA, Jaya, and SAMP-Jaya need 90000, 23925, 16870,18160, and 17,780 analyses, respectively. In comparison with the

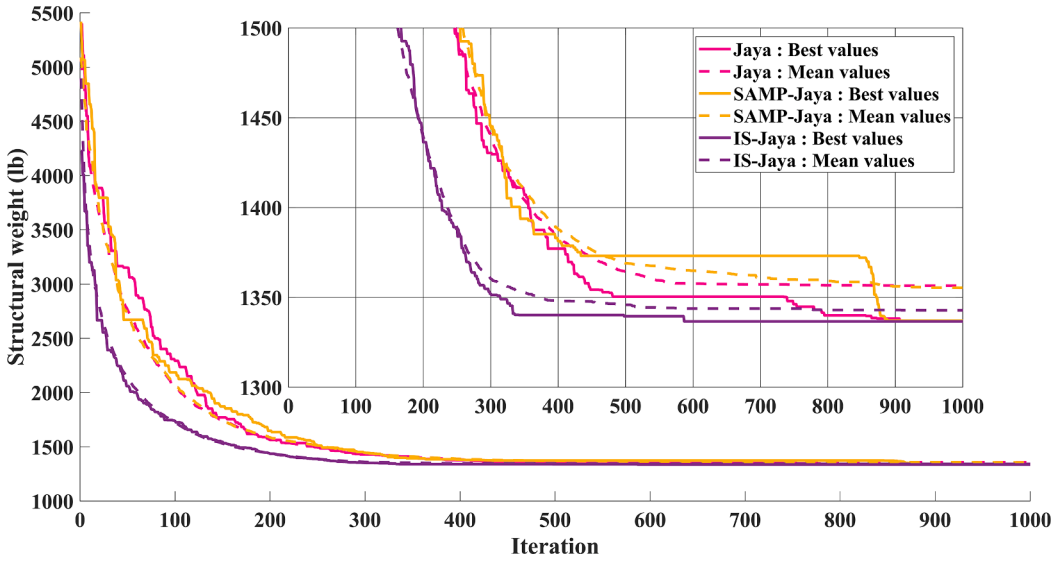

Fig. 17. Convergence history of the Jaya, SAMP-Jaya, and IS-Jaya for the 942-bar spatial truss structure.

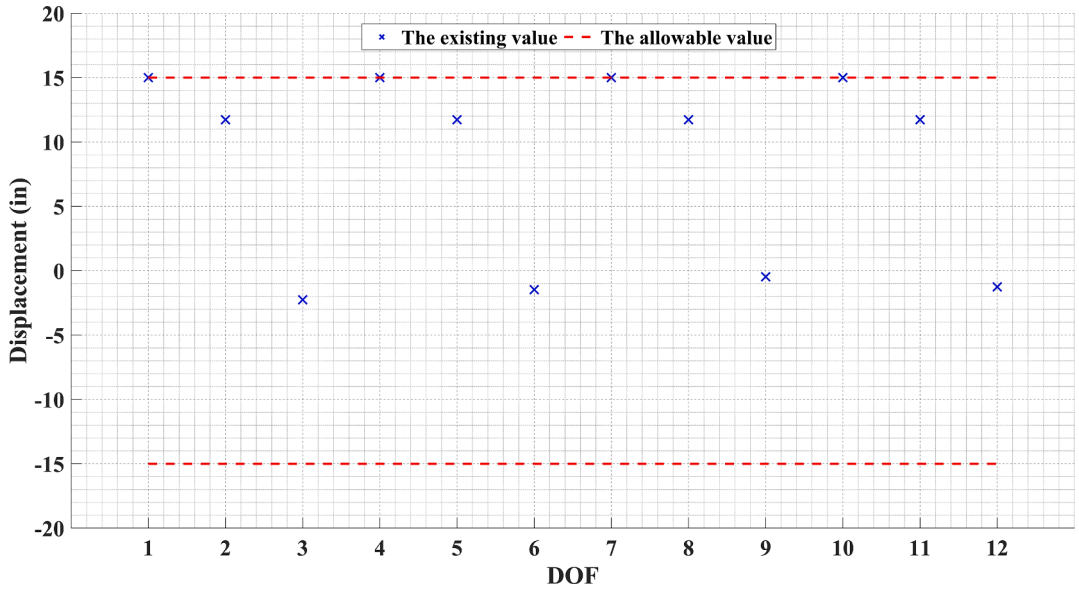

Fig. 18. Displacement values of top 4 nodes calculated at the obtained optimal design by the IS-Jaya for the 942-bar spatial truss structure.

average, worst, and standard deviation, the IS-Jaya algorithm is considerably much better than all other considered optimization methods. It can be concluded that IS-Jaya has better performance than the other existing methods reported in Table 8 in both aspects of accuracy and convergence speed. It is worth mentioning that this benchmark problem has been investigated by Kaveh et al. [51] as well. Nevertheless, the results obtained by them have a slight violation, so their results are not included in Table 8.

The optimum weight acquired by the IS-Jaya algorithm in each independent run is displayed in Fig. 12. As observed, the algorithm can gain the best weight (1336.634 kg) in 3 runs. According to this figure, only 6 runs found the optimum weight more than the average value. Convergence histories of the best and mean performance recorded for the three optimization methods namely Jaya, SAMP-Jaya, and IS-Jaya are compared in Fig. 13. This figure reveals that the curves of the IS-Jaya algorithm in both best and mean performance runs are below those of the Jaya and SAMP-Jaya algorithms during all iterations. Fig. 14 illustrates that the constraints of the problem have not been violated.

6.3. A 942-bar tower truss structure

In the third design example, the 942-bar spatial truss structure illustrated in Fig. 15 is studied to indicate the effectiveness and capability of the IS-Jaya algorithm in dealing with a problem that has a large number of design variables. The structure consists of 942 members and 244 nodes. As shown in Fig. 15, all elements are grouped into 59 element groups due to structural symmetry like the previous examples. The material density and modulus of elasticity are $\rho = 0.1lb/in^3$ and $E = 10,000$ ksi, respectively. A single loading condition imposed to the tower comprises of both lateral and vertical loads which are as follows:

(a) the vertical loads in the z-direction at each node in the first, second, and third sections are respectively equal to −3 kips, −6 kips, and −9 kips.
(b) the lateral loads in the y-direction at all nodes of the tower are equal to 1.0 kips.

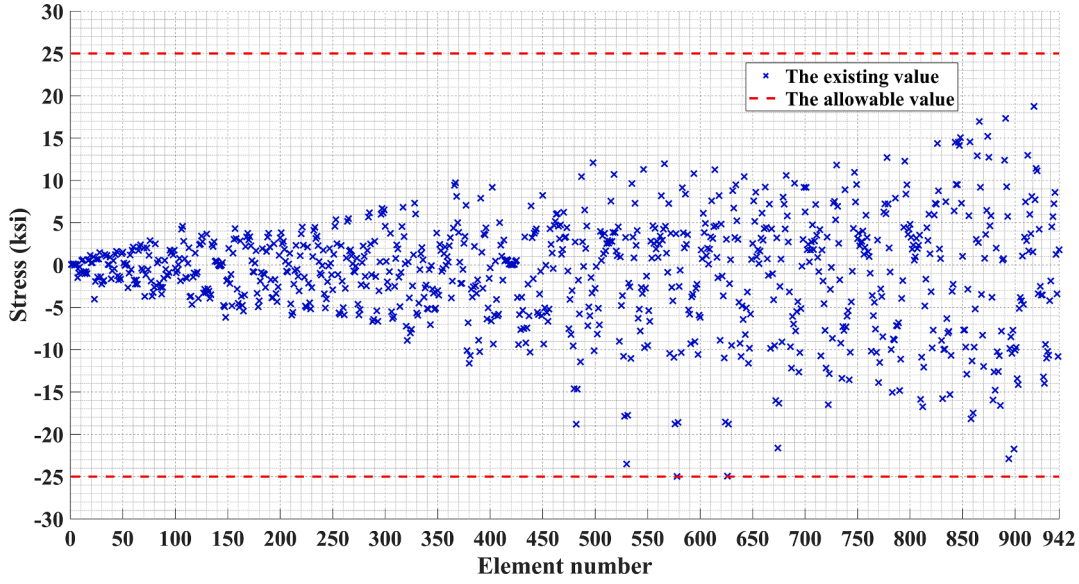

Fig. 19. Stress values calculated at the obtained optimal design by the IS-Jaya for the 942-bar spatial truss structure.

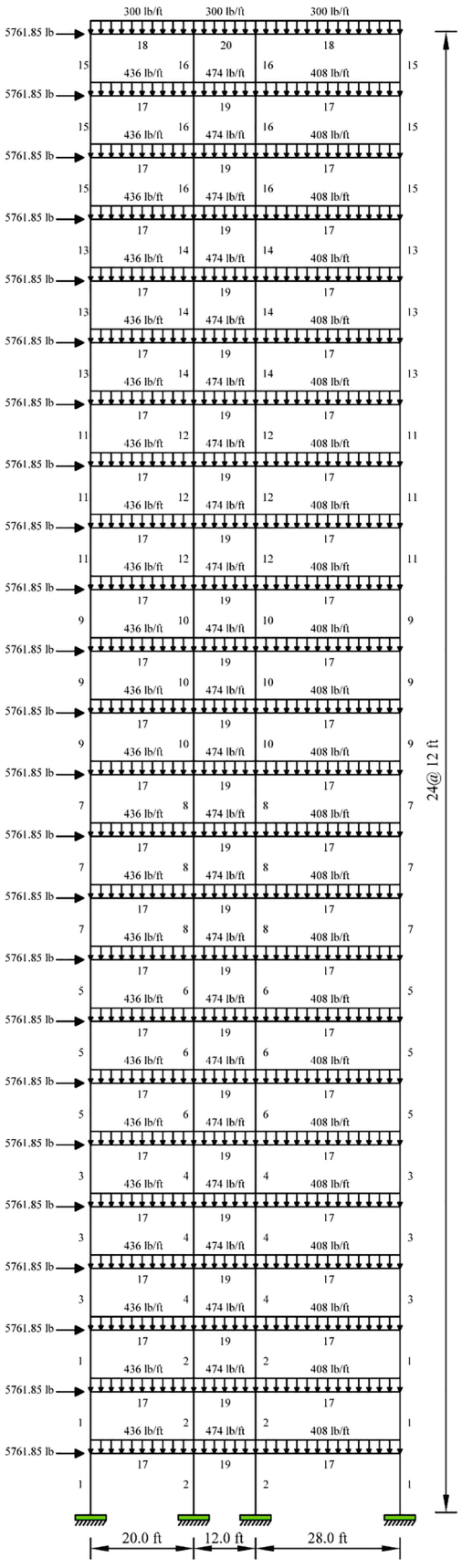

Fig. 20. The 3-bay 24-story steel frame.

(c) the lateral loads in the x-direction at each node on the left and right sides of the tower are respectively equal to 1.5 kips and 1.0 kips.

The cross-sectional areas are integer value so that they are selected from a range with lower bound 1.0 in.^{2} and upper bound 200 in.^{2}. The obtained stress value in both tension and compression members must not exceed 25.0 ksi. Furthermore, the nodal displacements of the four nodes on the top level in all directions must be smaller than the maximum allowable value which is 15 in.

Table 9 compares the results obtained by the IS-Jaya algorithm to those other optimization methods. The weight of the best optimum result found by IS-Jaya is 138688.913 lb which is the best among the compared methods. Also, the average, worst, and standard deviation of the results obtained by the IS-Jaya algorithm are respectively 142903.207 lb, 150721.603 lb, and 3171.403 lb, which are correspondingly less than all other methods. IS-Jaya algorithm requires 53,420 structural analyses to complete the optimization process, whereas Jaya and SAMP-Jaya need 55,700 and 56,920 analyses, respectively. On the other hand, although the NSAS in the IS-Jaya is more than CGFA and FA, the presented IS-Jaya algorithm has found the best optimum weight. This reveals that escaping from local optima incorporated in the body of the IS-Jaya works well, which leads to finding the best weight among the other optimization methods.

Fig. 16 illustrates all final results of 20 independent runs. The average and the best convergence histories recorded for Jaya, SAMP-Jaya, and IS-Jaya optimization algorithms are depicted in Fig. 17. This figure indicates that the convergence histories of IS-Jaya are significantly better than two other investigated methods in both aspects of accuracy and convergence rate. Figs. 18 and 19 show that both displacement and stress constraints of the problem have not been violated.

6.4. A 3-bay 24-story frame structure

A 3-bay 24-story steel frame structure depicted in Fig. 20 consists of 168 members (96 columns and 72 beams) is studied here as the last design example. The frame was originally designed by Davison and Adams [54]. This structure is one of the most popular examples in the field of structural optimization with discrete variables. All elements of the structure are classified into 20 distinct element groups (16 column groups and 4 beam groups). 16 column groups are chosen from W 14 shapes, while 4 beam groups are selected from all 267 W sections. The material has a modulus elasticity equal to $E = 29{,}732$ksi, and the yield stress is $F_y = 33.4$ ksi. The effective length factors of the members are calculated as $k_x \geq 0$ for a sway-permitted frame, and the out-of-plane effective length factor is determined as $k_y = 1.0$. All columns and beams are considered as non-braced along their lengths. In this example, the strength and displacement constraints are obtained according to the provisions of the AISC-LRFD requirements. These constraints contain:

(a) the maximum lateral displacements

$$\frac{\Delta_T}{H} - R \leq 0 \tag{12}$$

in which Δ_T is the maximum lateral displacement; H is the height of the structure, and R is maximum drift index which is equal to 1/300.

(b) the inter-story displacements

$$\frac{d_i}{h_i} - R_i \leq 0; i = 1, 2, \cdots, ns \tag{13}$$

where d_i and h_i are respectively the inter-story drift and the story height of the ith story;ns is the total number of the stories, and R_i denotes the allowable inter-story drift index of the ith story.

Table 10
Comparison results of different optimization methods for the 3-bay 24-story frame structure.

Element group	Optimal cross-sectional areas							
	HS [56]	ES-DE [58]	Accelerated WEO [45]	IBH [57]	CS [59]	Jaya	SAMP-Jaya	IS-Jaya
1	W14 × 176	W14 × 145	W14 × 159	W14 × 132	W14 × 176	W14 × 145	W14 × 159	W14 × 145
2	W14 × 145	W14 × 99	W14 × 132	W14 × 99	W14 × 109	W14 × 132	W14 × 109	W14 × 132
3	W14 × 176	W14 × 109	W14 × 99	W14 × 109	W14 × 99	W14 × 99	W14 × 132	W14 × 99
4	W14 × 132	W14 × 132	W14 × 109	W14 × 109	W14 × 99	W14 × 82	W14 × 90	W14 × 90
5	W14 × 132	W14 × 99	W14 × 68	W14 × 109	W14 × 74	W14 × 48	W14 × 61	W14 × 61
6	W14 × 109	W14 × 109	W14 × 38	W14 × 99	W14 × 38	W14 × 43	W14 × 38	W14 × 53
7	W14 × 109	W14 × 145	W14 × 30	W14 × 90	W14 × 30	W14 × 30	W14 × 34	W14 × 38
8	W14 × 82	W14 × 68	W14 × 22	W14 × 90	W14 × 22	W14 × 22	W14 × 22	W14 × 22
9	W14 × 82	W14 × 109	W14 × 90	W14 × 68	W14 × 90	W14 × 99	W14 × 90	W14 × 99
10	W14 × 61	W14 × 68	W14 × 99	W14 × 74	W14 × 109	W14 × 99	W14 × 109	W14 × 99
11	W14 × 74	W14 × 48	W14 × 99	W14 × 53	W14 × 99	W14 × 99	W14 × 90	W14 × 99
12	W14 × 48	W14 × 68	W14 × 74	W14 × 53	W14 × 82	W14 × 90	W14 × 82	W14 × 82
13	W14 × 34	W14 × 38	W14 × 68	W14 × 30	W14 × 68	W14 × 90	W14 × 74	W14 × 74
14	W14 × 30	W14 × 61	W14 × 61	W14 × 38	W14 × 61	W14 × 61	W14 × 61	W14 × 53
15	W14 × 22	W14 × 30	W14 × 34	W14 × 22	W14 × 38	W14 × 38	W14 × 34	W14 × 30
16	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 26	W14 × 22	W14 × 22
17	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W30 × 90
18	W10 × 22	W21 × 55	W8 × 18	W6 × 16	W6 × 15	W8 × 18	W8 × 18	W6 × 15
19	W18 × 40	W21 × 48	W24 × 55	W24 × 55	W24 × 55	W24 × 55	W24 × 55	W24 × 55
20	W12 × 16	W10 × 45	W6 × 8.5	W6 × 9	W6 × 8.5	W6 × 8.5	W6 × 8.5	W6 × 8.5
Best weight (lb)	214,860	212479.17	202194.02	208,719	202,482	202122.024	202410.034	201618.034
NSAs	13,924	12,500	11,300	10,000	18,760	17,660	16,600	14,100
Worst weight (lb)	N/A	N/A	N/A	N/A	N/A	219762.021	230940.004	209586.084
Mean weight (lb)	222,620	N/A	203412.88	215,226	230,342	206811.512	207979.482	203169.205
Standard deviation (lb)	N/A	N/A	N/A	N/A	65,703	4189.083	7031.549	1877.119

(c) strength constraint

$$\frac{P_u}{2\phi_c P_n}+\left(\frac{M_{ux}}{\phi_b M_{nx}}+\frac{M_{uy}}{\phi_b M_{ny}}\right)-1\le 0; if\ \frac{P_u}{\phi_c P_n}<0.2 \tag{14}$$

$$\frac{P_u}{\phi_c P_n}+\frac{8}{9}\times\left(\frac{M_{ux}}{\phi_b M_{nx}}+\frac{M_{uy}}{\phi_b M_{ny}}\right)-1\le 0; if\ \frac{P_u}{\phi_c P_n}\ge 0.2$$

where P_u is the required strength (tension or compression); ϕ_c is the resistance factor ($\phi_c = 0.9$ for tension, $\phi_c = 0.85$ for compression); M_{ux} and M_{uy} are respectively the required flexural strength in the x and y directions, while M_{nx} and M_{ny} are the nominal flexural strength in the x and y directions, respectively. ϕ_b is the flexural resistance reduction factor ($\phi_b = 0.90$). It should be noted that M_{ny} is equal to zero due to a two-dimensional structure.

The nominal tensile strength for yielding in the gross section is computed as:

$$P_n = A_g \times F_y \tag{15}$$

in which A_g is the gross cross-sectional area of the member and F_y is specified minimum yield stress. The nominal compressive strength of a member is calculated by the following equations:

$$P_n = A_g \times F_{cr} \tag{16}$$

$$F_{cr} = \left(0.658^{\lambda_c^2}\right) \times F_y; for \lambda_c \le 1.5 \tag{17}$$

$$F_{cr} = \left(\frac{0.877}{\lambda_c^2}\right) \times F_y; for \lambda_c > 1.5 \tag{18}$$

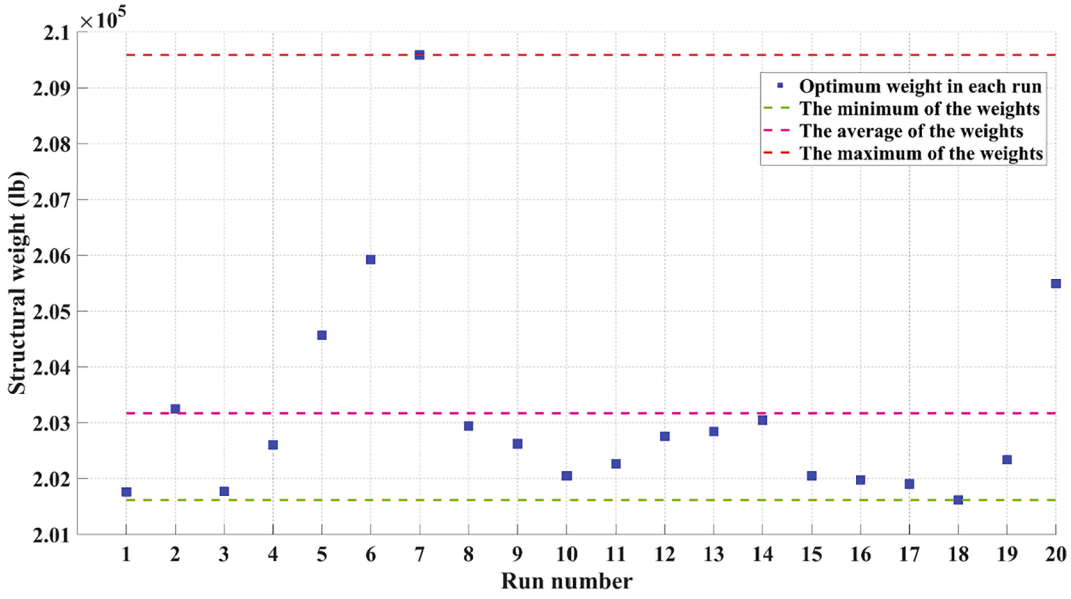

Fig. 21. The obtained structural weight in each independent run for the 3-bay 24-story frame structure.

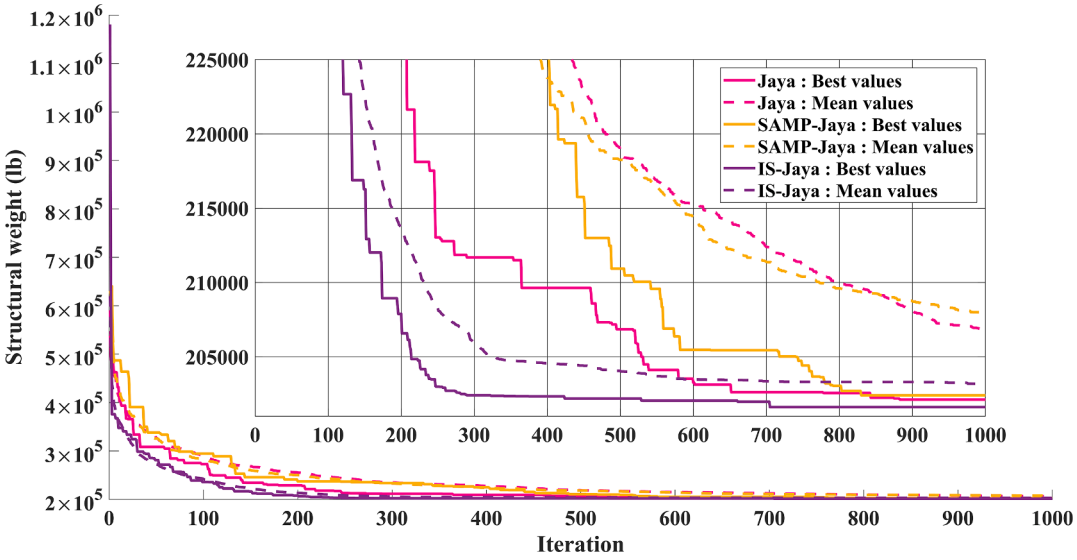

Fig. 22. Convergence history of the Jaya, SAMP-Jaya, and IS-Jaya for the 3-bay 24-story frame structure.

$$\lambda_c = \frac{kl}{r\pi} \times \sqrt{\frac{F_y}{E}} \tag{19}$$

where F_{cr} is the critical stress of the member, F_e is the elastic buckling stress of the member, E is the modulus of elasticity,r is the radius of gyration, L is laterally unbraced length of the member, and k is the effective length factor which can be calculated as follows:

$$k = \sqrt{\frac{1.6G_AG_B + 4 \times (G_A + G_B) + 7.5}{G_A + G_B + 7.5}} \tag{20}$$

where G_A and G_B are the stiffness ratios of columns and girders at two end joints, A and B, of the column section being considered, respectively [55].

The results obtained by the presented method and other existing optimization methods are compared in Table 10. From this table, it can be seen that the IS-Jaya algorithm has obtained the lightest weight (201618.034 lb) among all other methods namely HS [56] with a weight of 214860 lb, ES-DE [15] with a weight of 212479.17 lb, Accelerated WEO [5] with a weight of 202194.02 lb, IBH [57] with a weight of 208719, CS [16] with a weight of 202482 lb, Jaya with a weight of 202122.024 lb, and SAMP-Jaya with a weight of 202410.034 lb. Furthermore, the lowest average weight (203169.205 lb) has been acquired by the proposed method. This demonstrates that proposed IS-Jaya has the best performance than other methods in terms of accuracy. Moreover, according to this table, the IS-Jaya algorithm requires 14,100 structural analyses to find the best optimum weight (201618.034 lb), while Jaya and SAMP-Jaya algorithms respectively need 17,660 and 16,600 analyses.

All final results of 20 independent runs are given in Fig. 21. As can be seen, 15 independent runs out of 20 ones are less than the average of runs (203169.205 lb) which indicates the stability and robustness of the proposed method. Convergence histories of the best and average of runs recorded for the Jaya, SAMP-Jaya, and IS-Jaya algorithms are depicted in Fig. 22. As clear, the proposed method has much better performance than other methods in both aspects of computational cost and convergence speed. The inter-story drift which is obtained by the present method is illustrated in Fig. 23. The maximum inter-story drift is 0.4799

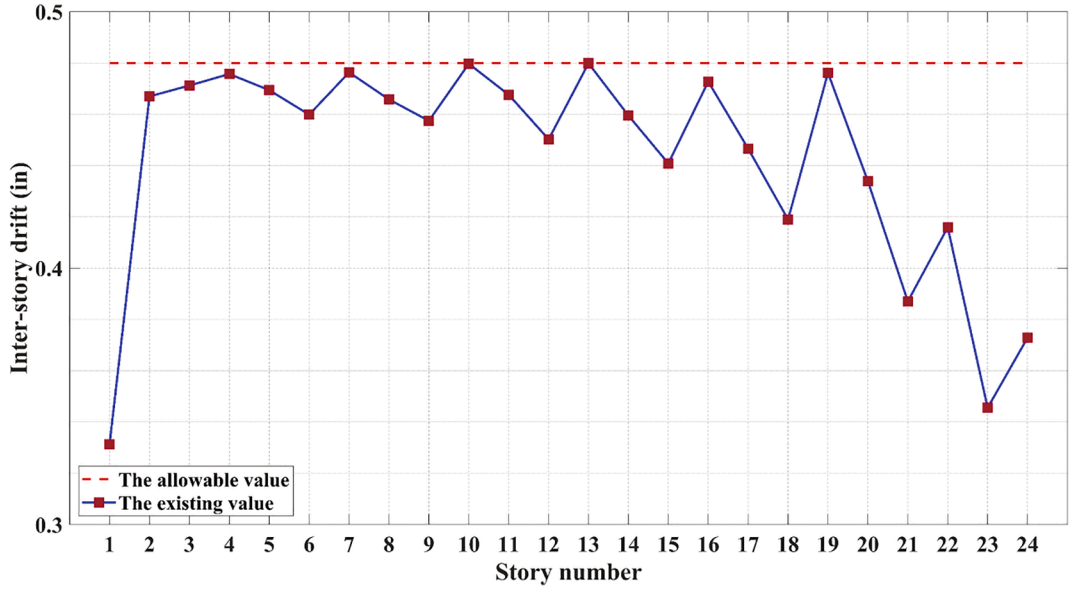

Fig. 23. Inter-story drift values calculated at the obtained optimal design by the IS-Jaya for the 3-bay 24-story frame structure.

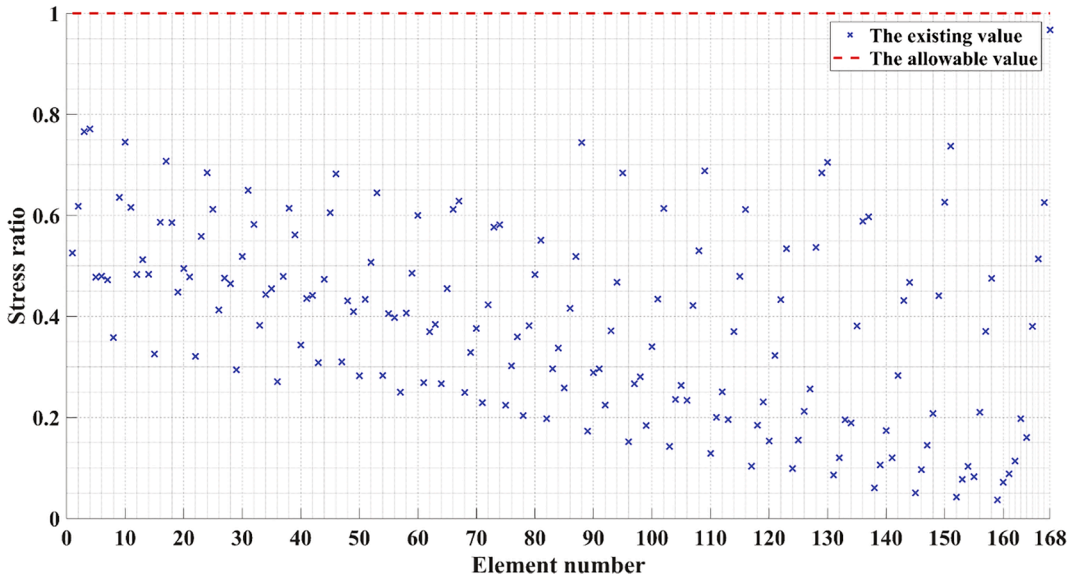

Fig. 24. Stress ratio values calculated at the obtained optimal design by the IS-Jaya for the 3-bay 24-story frame structure.

in. which is much close to its allowable value (0.48 in.). Fig. 24 provides the obtained stress ratio of the frame members. According to this figure, the maximum stress ratio is equal to 96.71% which happens in element 168.

7. Conclusions

This paper presented a new variant of Jaya algorithm namely Improved Shuffled based Jaya (IS-Jaya) algorithm. In the proposed method, the concept of the shuffling process and a mechanism to escape from local minima have been incorporated simultaneously into the standard version of the Jaya algorithm. In the IS-Jaya algorithm, the entire population is sorted based on the quality of solutions. Then, the population is partitioned into fixed subpopulations numbers, denoted as communities. In the proposed optimization method, the communities are formed through the process of shuffling. Each community is permitted to evolve independently by finding promising solutions based on the standard Jaya algorithm. This strategy makes to improve the solution by sharing the information independently acquired by each community. To avoid trapping into the local minima and improve the performance of the original Jaya algorithm, a mechanism is utilized to escape from local optima in each community. Finally, the communities share their information via merging to avoid unwanted premature convergence and also maintain population diversity.

The performance of the proposed method was successfully tested on four well-known discrete structural optimization problems. These examples include a 72-bar spatial truss structure, a 160-bar transmission tower structure, a 942-bar tower structure, and a 3-bay 24-story frame structure. In the first and second examples, although the optimum weight is available and IS-Jaya can find this weight like the other optimization methods, the IS-Jaya algorithm requires a smaller number of structural analyses than the other methods. This reveals the robustness of the proposed IS-Jaya not only in accuracy but also in convergence speed. In the third and last design examples, the present algorithm can find the best optimum weight among the other methods. However, the proposed IS-Jaya algorithm requires a bit larger number of structural analyses than SA, FA, and CGFA in the third design example and HS, ES-DE, Accelerated WEO, and IBH in the last design example. An increasing number of required structural analyses of the IS-Jaya algorithm due to adding a mechanism to escape from local optima lead to finding a better optimum weight than the other existing metaheuristic algorithms reported in this paper. On the other hand, in all investigated design examples, both accuracy and convergence speed of the proposed optimization method are consistently performing better than Jaya and SAMP-Jaya algorithms. In a general view, the optimization results reveal that the proposed IS-Jaya algorithm has significantly better performance than other existing methods. Moreover, the results obtained by the proposed IS-Jaya algorithm illustrates the superiority of the proposed method in comparison to Jaya and SAMP-Jaya algorithms.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Kaveh A, Ilchi Ghazaan M. Meta-heuristic algorithms for optimal design of real-size structures. Switzerland: Springer; 2018.
[2] Holland JH. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press; 1992.
[3] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[4] Duan Q, Gupta VK, Sorooshian S. Shuffled complex evolution approach for effective and efficient global minimization. J Optim Theory Appl 1993;76(3):501–21.
[5] Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of ICNN'95-International Conference on Neural Networks 1995;4:1942–8.
[6] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybernetics Part B (Cybernetics) 1996;26(1):29–41.
[7] Karaboga D. An idea based on honey bee swarm for numerical optimization, Technical report-tr06, Erciyes University, Engineering Faculty. Computer Engineering Department 2005;200:1–10.
[8] Eusuff M, Lansey K, Pasha F. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Eng Optim 2006;38(2):129–54.
[9] Yang X-S, Deb S. Cuckoo search via Lévy flights. World congress on nature & biologically inspired computing (NaBIC) 2009:210–4.
[10] Das S, Suganthan PN. Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 2010;15(1):4–31.
[11] Rao RV, Savsani VJ, Vakharia D. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 2011;43(3):303–15.
[12] Yang XS, Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. Engineering computations 2012;29(5):464–83.
[13] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61.
[14] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mech 2010;213(3–4):267–89.
[15] Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization. Comput Struct 2012;112:283–94.

[16] Kaveh A, Farhoudi N. A new optimization method: Dolphin echolocation. Adv Eng Softw 2013;59:53–70.
[17] Kaveh A, Mahdavi VR. Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 2014;139:18–27.
[18] Kaveh A, Bakhshpoori T. Water evaporation optimization: a novel physically inspired optimization algorithm. Comput Struct 2016;167:69–85.
[19] Kaveh A, Dadras Eslamlou A. A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv Eng Softw 2017;110:69–84.
[20] Kaveh A, Zaerreza A. Shuffled shepherd optimization method: a new Meta-heuristic algorithm. Eng Comput 2020;37(7):2357–89.
[21] Kaveh A, Dadras Eslamlou A. Water strider algorithm: A new metaheuristic and applications. Structures 2020;25:520–41.
[22] Ma H, Shen S, Yu M, Yang Z, Fei M, Zhou H. Multi-population techniques in nature inspired optimization algorithms: a comprehensive survey. Swarm Evol Comput 2019;44:365–87.
[23] Rao RV, Saroj A. A self-adaptive multi-population based Jaya algorithm for engineering optimization. Swarm Evol Comput 2017;37:1–26.
[24] Rao RV. Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int J Industrial Eng Comput 2016;7(1):19–34.
[25] Farah A, Belazi A. A novel chaotic Jaya algorithm for unconstrained numerical optimization. Nonlinear Dyn 2018;93(3):1451–80.
[26] Ghavidel S, Azizivahed A, Li L. A hybrid Jaya algorithm for reliability–redundancy allocation problems. Eng Optim 2018;50(4):698–715.
[27] Wang L, Huang C. A novel Elite Opposition-based Jaya algorithm for parameter estimation of photovoltaic cell models. Optik 2018;155:351–6.
[28] Aslan M, Gunduz M, Kiran MS. JayaX: Jaya algorithm with xor operator for binary optimization. Appl Soft Comput 2019;82:105576.
[29] Degertekin S, Lamberti L, Ugur I. Discrete sizing/layout/topology optimization of truss structures with an advanced Jaya algorithm. Appl Soft Comput 2019;79:363–90.
[30] Ding Z, Li J, Hao H. Structural damage identification using improved Jaya algorithm based on sparse regularization and Bayesian inference. Mech Syst Sig Process 2019;132:211–31.
[31] Migallón H, Jimeno-Morenilla A, Sánchez-Romero J-L, Rico H, Rao RV. Multipopulation-based multi-level parallel enhanced Jaya algorithms. J Supercomputing 2019;75(3):1697–716.
[32] Nayak DR, Zhang Y, Das DS, Panda S. MJaya-ELM: a Jaya algorithm with mutation and extreme learning machine based approach for sensorineural hearing loss detection. Appl Soft Comput 2019;83:105626.
[33] Rao RV, Saroj A. An elitism-based self-adaptive multi-population Jaya algorithm and its applications. Soft Comput 2019;23(12):4383–406.
[34] Ingle KK, Jatoth RK. An efficient JAYA algorithm with Lévy flight for non-linear channel equalization. Expert Syst Appl 2020;145:112970.
[35] Ho-Huu V, Nguyen-Thoi T, Vo-Duy T, Nguyen-Trang T. An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. Comput Struct 2016;165:59–75.
[36] Yates D, Templeman A, Boffey T. The complexity of procedures for determining minimum weight trusses with discrete member sizes. Int J Solids Struct 1982;18(6):487–95.
[37] Stolpe M. Truss optimization with discrete design variables: a critical review. Struct Multidiscip Optim 2016;53(2):349–74.
[38] Rao RV. Jaya: an advanced optimization algorithm and its engineering applications. Springer; 2019.
[39] Wu G, Mallipeddi R, Suganthan PN, Wang R, Chen H. Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 2016;329:329–45.
[40] Chaudhary R, Banati H. Study of population partitioning techniques on efficiency of swarm algorithms. Swarm Evol Comput 2020:100672.
[41] Chaudhary R, Banati H. Swarm bat algorithm with improved search (SBAIS). Soft Comput 2019;23(22):11461–91.
[42] Wu S-J, Chow P-T. Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 1995;56(6):979–91.
[43] Kaveh A, Mahdavi VR. Colliding bodies optimization method for optimum discrete design of truss structures. Comput Struct 2014;139:43–53.
[44] Sadollah A, Eskandar H, Bahreininejad A, Kim JH. Water cycle, mine blast and improved mine blast algorithms for discrete sizing optimization of truss structures. Comput Struct 2015;149:1–16.
[45] Kaveh A, Bakhshpoori T. An accelerated water evaporation optimization formulation for discrete optimization of skeletal structures. Comput Struct 2016;177:218–28.
[46] Jalili S, Hosseinzadeh Y. Design optimization of truss structures with continuous and discrete variables by hybrid of biogeography-based optimization and differential evolution methods. Struct Design Tall Special Buildings 2018;27(14):e1495.
[47] Le DT, Bui D-K, Ngo TD, Nguyen Q-H, Nguyen-Xuan H. A novel hybrid method combining electromagnetism-like mechanism and firefly algorithms for constrained design optimization of discrete truss structures. Comput Struct 2019;212:20–42.
[48] Groenwold A, Stander N. Optimal discrete sizing of truss structures subject to buckling constraints. Struct Optim 1997;14(2–3):71–80.
[49] Groenwold A, Stander N, Snyman J. A regional genetic algorithm for the discrete optimal design of truss structures. Int J Numer Meth Eng 1999;44(6):749–66.
[50] Capriles PV, Fonseca LG, Barbosa HJ, Lemonge AC. Rank-based ant colony algorithms for truss weight minimization with discrete variables. Commun Numer Methods Eng 2007;23(6):553–75.
[51] Kaveh A, Kalatjari VR, Talebpour MH. Optimal design of steel towers using a multi-metaheuristic based search method. Periodica Polytechnica Civil Engineering 2016;60(2):229–46.
[52] Hasancebi O, Erbatur F. On efficient use of simulated annealing in complex structural optimization problems. Acta Mech 2002;157(1–4):27–50.
[53] Kaveh A, Moghanni RM, Javadi S. Optimum design of large steel skeletal structures using chaotic firefly optimization algorithm based on the Gaussian map. Struct Multidiscip Optim 2019;60(3):879–94.
[54] Davison JH, Adams PF. Stability of braced and unbraced frames. J Struct Division 1974;100(2):319–34.
[55] Dumonteil P. Simple equations for effective length factors. Eng J AISC 1992;29(3):111–5.
[56] Degertekin SO. Optimum design of steel frames using harmony search algorithm. Struct Multidiscip Optim 2008;36(4):393–401.
[57] Gholizadeh S, Razavi N, Shojaei E. Improved black hole and multiverse algorithms for discrete sizing optimization of planar structures. Eng Optim 2019;51(10):1645–67.
[58] Talatahari S, Gandomi AH, Yang X-S, Deb S. Optimum design of frame structures using the eagle strategy with differential evolution. Eng Struct 2015;91:16–25.
[59] Kaveh A, Hamedani KB, Hosseini SM, Bakhshpoori T. Optimal design of planar steel frame structures utilizing meta-heuristic optimization algorithms. Structures 2020;25:335–46.