Complex & Intelligent Systems (2018) 4:195–212
https://doi.org/10.1007/s40747-018-0066-z

ORIGINAL ARTICLE

A hybrid SA-MFO algorithm for function optimization and engineering design problems

Gehad Ismail Sayed[1,2] · Aboul Ella Hassanien[1,2]

Received: 10 August 2017 / Accepted: 5 January 2018 / Published online: 25 January 2018

Abstract
This paper presents a hybrid algorithm based on using moth-flame optimization (MFO) algorithm with simulated annealing (SA), namely (SA-MFO). The proposed SA-MFO algorithm takes the advantages of both algorithms. It takes the ability to escape from local optima mechanism of SA and fast searching and learning mechanism for guiding the generation of candidate solutions of MFO. The proposed SA-MFO algorithm is applied on 23 unconstrained benchmark functions and four well-known constrained engineering problems. The experimental results show the superiority of the proposed algorithm. Moreover, the performance of SA-MFO is compared with well-known and recent meta-heuristic algorithms. The results show competitive results of SA-MFO concerning MFO and other meta-heuristic algorithms.

Keywords Simulated annealing algorithm · Moth flame optimization algorithm · Function optimization · Constrained optimization problems

Introduction

Optimization is defined as the process of finding global or near global optimum solutions for a given problem. Many problems in the real world can be observed as optimization problems. Over the past several decades, several optimization algorithms are proposed to solve many optimization problems. Simulated annealing (SA) algorithm is one of the most popular iterative meta-heuristic optimization algorithm. It used to solve continuous and discrete problems [16]. The key privilege of SA is that its ability to escape from local optima called metropolis process. However, the major shortage of SA is that its efficiency for finding the global optimum is unsatisfactory. This is due to, when SA generates a new candidate solution, it does not learn intelligently from its searching history [37]. Several studies have been introduced for enhancing SA performance. Some of these are based on changing the acceptance or generation mechanisms. However, this kind of modifications in SA schema does not usually inherit the ability of SA for escaping from local optima [11]. On the other hand, population-based meta-heuristics algorithms equipped with the ability to guide their search through using intelligent learning mechanism. Thus, the population-based meta-heuristics algorithms showed better efficiency. Some of these algorithms are particle swarm optimisation (PSO) [15], ant colony optimization (ACO) [4], ant lion optimizer (ALO) [22], differential evolution (DE) [35], Social group optimization (SGO) [30], and moth-flame optimization (MFO) [24].

In this paper, MFO is selected to be studied and analyzed. MFO algorithm is one of recent meta-heuristic optimization algorithms proposed in 2015. The main inspiration of MFO came from the navigation method of moths in nature called transverse orientation. The inventor of this algorithm, Mirjalili, showed that MFO obtained very competitive results compared with other meta-heuristic optimization algorithms. Also, MFO shows competitive results for automatic mitosis detection in breast cancer histology images [32]. However, MFO like other meta-heuristic algorithms suffers from entrapping at local optima and low convergence rate. Thus, many studies are proposed to solve this problem. Several studies have been proposed to enhance the performance of MFO. Li et al. [17] proposed Lévy-flight moth-flame optimization (LMFO) algorithm to improve the performance

✉ Gehad Ismail Sayed
GehadIsmail_FCI@yahoo.com

Aboul Ella Hassanien
aboitcairo@gmail.com

1 Faculty of Computers and Information, Cairo University, Cairo, Egypt

2 Scientific Research Group in Egypt (SRGE), Cairo, Egypt

of MFO. The proposed LMFO used Lévy-flight strategy in the searching mechanism of MFO. The experimental results on 19 unconstrained benchmark functions and two other constrained problems showed the superior performance of LMFO. Emery et al. [5] employed chaos parameter in the spiral equation of updating the position of moths. They applied their approach in feature selection application. Recently, many studies hybridize two or more algorithms to obtain optimal solutions for optimization problems [31]. Some of these studies hybridize differential evolution with biogeography-based optimization to solve global optimization problem [8]. Some hybridize particle swarm optimization with differential evolution for solving constrained numerical and engineering optimization problems [19]. Moreover, some of them hybridize chaotic with meta-heuristic algorithms for solving feature selection problem [33]. Also, some hybridize genetic algorithm and particle swarm optimization for multimodal problems [13]. Furthermore, some researchers considering introducing new methods to improve meta-heuristic algorithms. Author in [25] introduces new mutation rule to be combined with the basic mutation strategy of differential evolution algorithm. He named his algorithm as enhanced adaptive differential evolution (EADE). The experimental results show that EADE is very competitive algorithm for solving large-scale global optimization problems. Authors in [28] combined two binary bat algorithm (BBA) and with local search scheme (LSS)to improve the diversity of BBA and enhance the convergence rate. The results show that the proposed hybrid algorithm obtained promising results for solving large-scale 01 knapsack problems. Authors in [36] introduced a mutation operator guided by preferred regions to enhance the performance of an existing set-based evolutionary many-objective optimization algorithm. The results obtained from 21 instances of seven benchmark show that their proposed algorithm is superior. The main drivers for this kind of hybridization are to avoid local optima and improve search results from global optimization. This becomes an essential task in multidimensional and multiobjective problems.

Therefore, in this paper, we tried another kind of hybridization to solve the main problems of MFO. To best of our knowledge, this kind of hybridization of MFO with SA is not used before. The salient features of SA and MFO are hybrid to create a new approach, namely simulated annealing moth-flame algorithm (SA-MFO). In this paper, simulated annealing method is applied during the updating positions of flames of the standard version of MFO to further enhance the performance of MFO. The proposed SA-MFO is applied to solve both constrained and unconstrained benchmark problems. The proposed mimetic algorithm is mainly based on MFO algorithm, whereas, metropolis mechanism of SA is used to slow down the convergence rate of MFO and increases the probability of moths to reach the global optima. Also, metropolis mechanism is used to increase the diversity of the population. The proposed SA-MFO has the advantages of both SA and MFO algorithms: the ability to escape from local optima mechanism of SA and fast searching ability and learning mechanism for guiding the generation of candidate solutions of MFO.

The proposed mimetic algorithm applied on 23 benchmark functions and four well-known engineering problems. These problems are applied previously on the original MFO. Thus, we almost make a fair comparison between the proposed SA-MFO and MFO in terms of evaluation criteria, number of iterations, number of independent runs, population size, and number of dimensionality. The simulation results demonstrate that the proposed SA-MFO can significantly improve the performance of MFO. Also, the experimental results show that SA-MFO is superior to the other algorithms.

The rest of this paper is organized as follows: Section 2 provides a brief overview of the standard moth flame optimization algorithm, followed by a brief introduction to simulated annealing. Section 3 describes in detail the proposed hybrid algorithm, and then Sect. 4 discusses the main results and evaluate the performance of the proposed algorithm. Finally, Sect. 5 concludes the paper.

Basics and background

Moth flame optimization (MFO)

Inspiration

In this section, one of the recent stochastic population-based algorithms is employed namely moth flame optimization (MFO) [24]. The main inspiration of MFO came from the navigation method of moths in nature. Moths are fancy insects that are very similar to the butterfly family. In nature, there are greater than 160,000 various species of this insect. Larvae and adult are the two main milestones in their lifetime. The larva is converted to moth by cocoons. Special navigation methods in the night are the most interesting fact about moths. They used a mechanism called transverse orientation for their navigation. Moths fly using a fixed angle with respect to the moon, which is a very efficient mechanism for long traveling distances in a straight line.

Mathematical model of MFO

Let the candidate's solutions are moths, and the problem's variables are the position of moths in the space. P is the spiral function where moths move around the search space. Each moth updates his position with respect to flame using the following equations:

$$M_i = P(M_i, F_j) \tag{1}$$

where M_i indicates the ith moth and F_j is j_th flame.
There are other types of spiral functions can be utilized respect to the following rules:

1. The initial point of spiral should start from the moth.
2. The final point of spiral should be the position of the flame.
3. The Fluctuation range of spiral shouldn't exceed the search space.

$$P(M_i, F_j) = D_i \cdot \mathrm{e}^{bt} \cdot \cos(2\pi t) + F_j, \tag{2}$$

where D_i is the distance of the ith moth for the jth flame, t is a random number in $[-1, 1]$ and b is a constant for defining the shape of the P. D is calculated using the following equation.

$$D_i = |F_j - M_i|, \tag{3}$$

where M_i is the ith moth, F_j indicates the jth flame, and D_i indicates the distance of the ith moth to the jth flame.
Another concern, the moths update their position with respect to n different locations in the search space which can degrade the best promising solutions exploitation. Therefore, the number of flames adaptively decreases over the course of iterations using the following formula:

$$\text{flame}_{\text{no}} = \text{round}\left(\text{FN} - I * \frac{N-1}{\text{I}N}\right), \tag{4}$$

where I is the current number of iterations, IN is the maximum number of iterations and FN is the maximum number of flames.
MFO utilizes Quicksort method and the computational complexity of this sort method is $O(nlogn)$ and $O(n^2)$ in the best and worst case, where n denotes the number of moths.

Simulated annealing (SA)

Simulated annealing (SA) was proposed by Metropolis et al. [21] in 1953. The main inspiration of SA came from the simulation modeling of a molecular movement in the materials during annealing (physical annealing principles). The process of cooling and heating a material to recrystallized is called annealing. At high temperature, the particles move in disorder and when the temperature decreases gradually, the particles converge to the lowest state of energy. SA uses probability acceptance mechanisms for considering the current optimal solution. SA was successfully applied to solve the optimization problems by Kirkpatrick et al. [15] in 1983. Through using the probability of acceptance of metropolis process, it can be simulated the annealing mechanism in finding the lowest energy state which is the optimum solution. The main parameters of SA are metropolis acceptance mechanism, initial condition, and cooling scheduling method. They are described as the following:

Initial condition at the beginning of the searching process of SA algorithm starts with initializing the initial temperature T_0, which have a significant impact on the performance. The higher value of T_0 will increase the computational time of the optimization process. On the contrary, if the T_0 is too low, it will cause that SA will not effectively explore the search space. Thus it is needed to be selected carefully to obtain the optimal solution.

Metropolis acceptance mechanism It is one of key affecting on obtaining a near optimal solution quickly. The metropolis rules used to indicate whether the nearby solution with low fitness value is accepted or not as the current solution. This mechanism affects the capability of SA to escape from the local optimum. Let F_C represents the fitness value of the current solution, and F_N denotes the fitness value of the neighboring solution. When $F(Y_0)$ is better than $F(Y)$, then SA will use the acceptance probability mechanism to indicating either to consider the neighbor solution as the current solution or not. The probability of acceptance mechanism is defined as follows:

$$\text{Pr} = \mathrm{e}^{\left(\frac{-(F(Y)-F(Y_0))}{T_k}\right)}, \tag{5}$$

where Pr is defined as the probability of acceptance and T_k is temperature value at time k.

Cooling schedule method During the physical annealing process, as the temperature decreases, the cooling rate decreases too to reach the stable ground state. This it's needed that the system at the beginning to be cooled faster and slower as gradually decreased of the temperature. The cooling schedule parameter is defined as follows:

$$T_{k+1} = \alpha \times T_k, \tag{6}$$

where α denotes the temperature coefficient, T_k is the initial temperature value and T_{k+1} is temperature at time k

The proposed SA-MFO algorithm

According to the MFO algorithm, moths' positions are randomly initialized. Then moth swarm updates their position movement using Eq. (2). Through this, the next moth position is determined. The standard version of MFO automatically accepts the current moth position as the new moth position. However, SA-MFO uses the metropolis acceptance mechanism of SA. This mechanism indicates whether to accept

Table 1 Parameters settings for SA-MFO Algorithm

Parameter	Value(s)
Number of search agents	50
Cooling schedule	10
Initial temperature	90
a	− 1
b	1

the new moth position or not. In case of the new moth, the position is not accepted, another candidate position is recalculated. The selection is based on its fitness value. Using this mechanism, a moth solution able to escape from the local optima. Also, the quality of the solution is enhanced with fastest convergence rate [34]. Based on temperature, cooling schedule parameters and the fitness value difference of the metropolis mechanism, the optimization process will explore the search space effectively to find an optimal solution. This process will be repeated until a new position is accepted using metropolis acceptance rule, or the termination criterion is reached.

Parameters initialization

In the beginning, SA-MFO starts with setting MFO parameters and randomly initialized moths' positions within the search space. Each position represents a solution in the search space. The initial parameter settings are presented in Table 1.

Fitness function

At each iteration, each moth position is evaluated using a predefined fitness function $f(\mathbf{z})$. Then based on this value, the metropolis rule is applied to determine whether to accept or not.

Updating positions

If the current moth accepted as the new moth, then the new moths change its position using Eq. (2).

Termination criteria

The optimization process terminates when it reaches the maximum number of iterations or when the best solution is found [7]. In our case, we used the maximum number of iterations is 1000 iterations (see Table 1).

The overall SA-MFO algorithm is proposed in Algorithm (1). Moreover, the summarize flowchart of the proposed simulated annealing moth-flame algorithm is given in Fig. 1.

Algorithm 1 Simulated Annealing Moth Flame Optimization Algorithm

1: Set the initial value of the number of moth positions n, T_0, α and the maximum number of iterations Max_{iter}.
2: **for** $(i = 1 : n)$ **do**
3: Assign randomly moths positions $\mathbf{M_i(t)}$.
4: Evaluate the fitness function of each moth position $f(\mathbf{M_i})$.
5: **end for**
6: Set $T = T_0$
7: **repeat**
8: Calculate $\Delta Fitness = f(\mathbf{M_{i+1}}) - f(\mathbf{M_i})$
9: $r = [0, 1]$.
10: **if** $f(\mathbf{M_{i+1}}) \leq f(\mathbf{M_i})$ **then**
11: Calculate D using Equ. (3).
12: Update $M(i, j)$ using Equ. (1) and (2) with respect to moth.
13: **else**
14: Set $Pr = e^{(\frac{-(F(Y)-F(Y_0))}{T})}$
15: **if** $r < Pr$ **then**
16: Calculate D using Equ. (3).
17: Update $M(i, j)$ using Equ. (1) and (2) with respect to moth.
18: **end if**
19: **end if**
20: Check if the new positions go out of the search space boundaries and bring it back.
21: Set $T = \alpha \times T$
22: Set $i = i + 1$. {Iteration counter increasing}.
23: **until** $(i < Max_{iter})$. {Termination criteria satisfied}.
24: Produce the best flame position and its fitness value.

Results and discussion

In this paper, 23 optimization benchmark problems with different characteristics and four other engineering problems are employed to evaluate the performance of the proposed SA-MFO algorithm. The employed 23 benchmark problems divided into two families: unimodal and multimodal. Tables 11 and 12 in Appendix 1 and 2 describes the mathematical formulation and the properties of the benchmark functions used in this paper. In the table, lb and ub represent the lower and upper bound of the search space respectively, dim indicates the number of dimensions, and opt indicates the optimum solution of the function. More information about the adopted benchmark functions can be found in [18]. The main characteristics of the first family, namely unimodal test functions have only one global optimum and no local optima. These functions are used to evaluate the exploitation and convergence rate of the adopted algorithms. However, the second family, namely multimodal test functions have multiple global optimum and multiple local optima. The characteristic of this family is used to evaluate the capability of the algorithm in avoiding the local optima and the explorative ability of an algorithm.

In this section, three main experiments were conducted to evaluate the performance of the SA-MFO algorithm. The first experiment aims to determine the optimal parameter value for both T_0 and α. The second experiment aims to evalu-

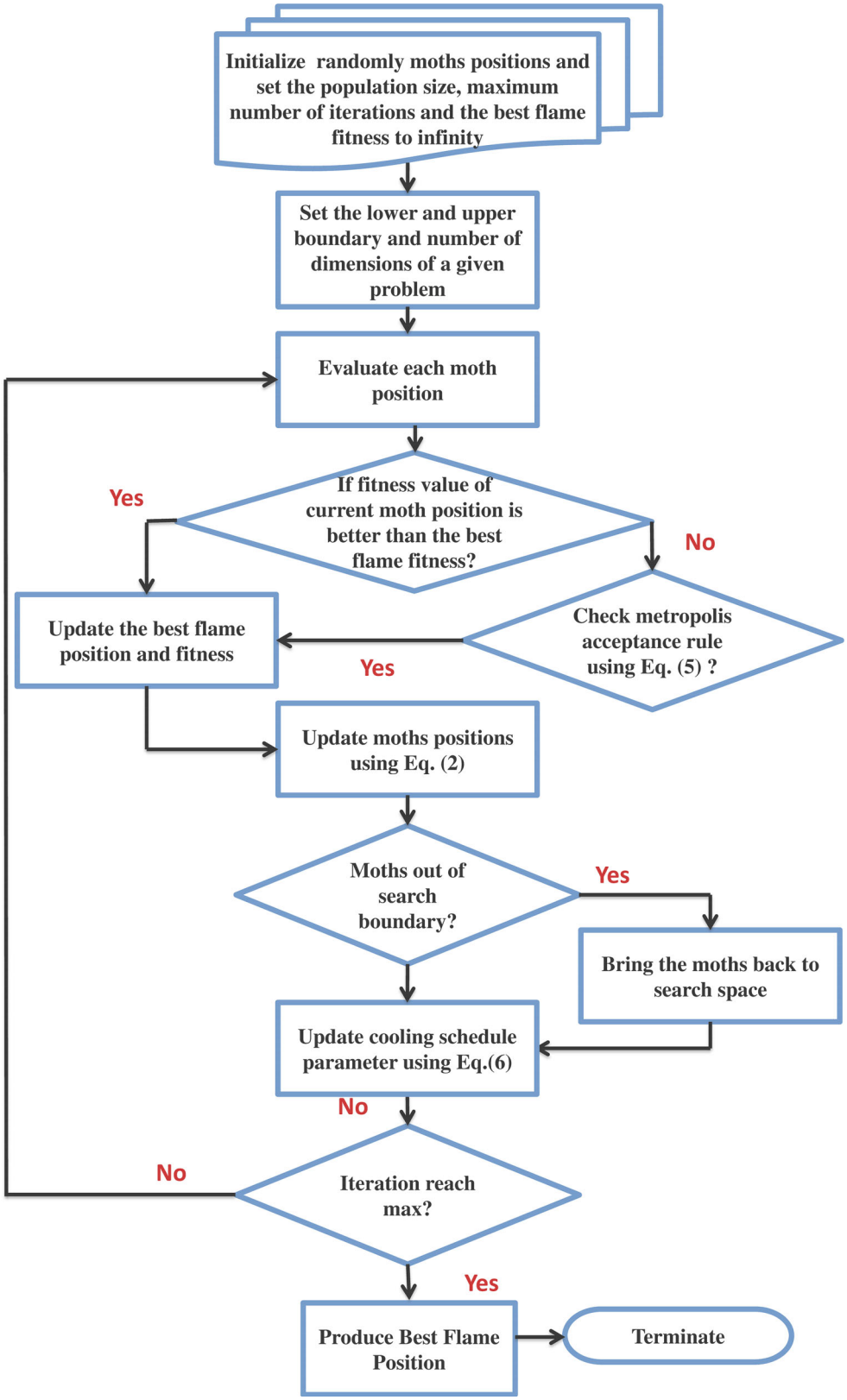


Fig. 1 SA-MFO algorithm flowchart

Table 2 Parameter settings for PSO, ABC, ALO, MFO, MVO, and GWO optimization algorithms

Algorithm	Parameter	Value
PSO	An inertial weight	1
	A inertia weight damping ratio	0.9
	Personal learning coefficient	1.5
	Global learning coefficient	2.0
ABC	A number of colony size	10
	A number of food source	5
	A number of limit trials	5
ALO	I	1
MFO	a	− 1
	b	1
MVO	WEP_{min}	0.2
	WEP_{min}	1
	TDR_{min}	0
	TDR_{max}	0.6
	p	6

ate and compare the performance of SA-MFO with MFO and five well-known optimization algorithms: particle swarm optimization (PSO) [15], ant bee colony (ABC) [14], moth flame optimization (MFO) [24], multi verse optimization (MVO) [23], and ant lion optimizer (ALO) [22] in solving 23 numerical optimization problems using different statistical measurements. The parameters settings for all meta-heuristic optimization algorithms are shown in Table 2. The rest of parameters including, the maximum number of iterations for all functions is set to 1000, the number of search agents is set to 50 individuals for all comparisons and the initialization of the search agents is same for all adopted algorithms. The reason behind this that we want to make almost a fair comparison for these algorithms. The third and last experiment aims to evaluate the performance of SA-MFO on solving engineering problems, also to compare the best-obtained results with other meta-heuristic algorithms. All the obtained results for both benchmark and engineering problems are calculated on average for 30 independent runs. Moreover, all these experiments are performed on the same PC with Core i3 and RAM 2 GB on OS windows 7.

Parameters optimization experiment

Due to the stochastic behavior of the optimization algorithms, there is a need to set certain parameters carefully. These parameters can have a significant influence on the performance of the algorithm. The main parameters of SA are T_0 and α which affecting the searching efficiency to find the optimal solution. These two parameters need to be optimized effectively. In this paper, the optimal parameters values for T_0 and α are identified based on the highest success rate for 50 runs of F_1, F_{10} and F_{18}. This paper uses the same mechanism proposed in [34] which depends on optimizing one parameter per simulation. When an optimal value of the parameter is determined, the optimization process is repeated till to find an optimal value of the next parameter. All the obtained results reach $1 \times E^{-3}$ are observed as a success.

Table 3 Parameter setting for initial acceptance rate T_0

Success rate (%)				
T_0 (%)	F1	F10	F18	AVG success rate (%)
10	100	90	79	89.67
30	100	91	76	89
50	100	91	80	90.33
70	100	92	75	89
90	**100**	**92**	**80**	**90.67**

The significance bold means indicates the optimal values for T_0, which are determined based on highest success rate for 50 runs of F1; F10 and F18

Table 4 Parameter setting for schedule rate α

Success rate (%)				
α (%)	F1	F10	F18	AVG success rate (%)
10	**100**	**92**	**99**	**97**
30	100	92	75	89
50	100	91	89	93.33
70	100	91	75	88.67
90	100	92	50	80.67

The significance bold means indicates the optimal values for α, which are determined based on highest success rate for 50 runs of F1; F10 and F18

Table 3 shows the simulation results obtained for selecting the optimal value of T_0. Changing the value of T_0 can highly affect on the initial acceptance rate of SA and SA-MFO. As, if the T_0 is too high that will lead the algorithm to spend much time in exploiting solution space and if the α is too low that will cause increasing the search time. In this paper, T_0 set with 10% and increases by 10% each time till to reach 90% as shown in Table 3. As it can be seen, T_0 with value 90% improves the success rate (Table 3).

α can highly affect by the acceptance of the poor solution which may lead to exploring the search space without finding an optimal solution. If α is too low, it may lead to stuck in the local optima. In this paper, α set with 10% and increases by 10% each time till to reach 90% as shown in (Table 4). As it can be seen, α with value 10% improves the success rate.

Comparison using numerical benchmark functions experiment

In this subsection, the performance of SA-MFO is analyzed and compared with other meta-heuristic algorithms in terms of mean, standard deviation (std.) for 30 independent runs.

Table 5 Statistical results of 23 benchmark functions obtained by SA-MFO, PSO, ALO, ABC, MFO, and MVO

	SA-MFO		PSO		ALO		ABC		MFO		MVO	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
F1	1.9E−31	4.6E−31	2.5E−20	1.5E−25	3.4E−14	1.8E−14	6.3E−13	1.0E−12	2.6E−31	5.5E−31	2.4E−03	1.3E−03
F2	3.0E−22	1.5E−21	4.0E−19	6.5E−19	2.0E−05	1.6E−05	5.5E−11	1.0E−10	2.4E−19	3.3E−19	1.4E−02	5.0E−03
F3	6.7E−04	8.8E−05	2.2E−04	4.8E−05	1.0E−04	2.3E−04	1.3E+01	8.0	7.5E−03	1.7E−03	5.9E−02	2.9E−02
F4	1.8E−03	6.0E−03	4.2E−01	1.2E−01	3.8E−03	3.7E−03	6.1E−01	2.6E−01	4.6E−02	1.5E−01	6.6E−02	1.9E−02
F5	1.5E+01	2.2E+01	2.1	1.7	5.1	3.1	1.7E+01	1.2E+01	1.2E+02	5.5E+02	8.8E+01	2.9E+02
F6	2.8E−14	3.4E−14	3.0E−12	3.0E−12	2.3E−09	1.1E−09	9.0E−13	1.1E−12	2.8E−13	3.4E−13	7.0E−03	3.0E−03
F7	4.2E−03	2.2E−03	9.3E−04	4.4E−04	6.8E−03	2.9E−03	5.4E−03	2.3E−03	4.6E−03	2.5E−03	1.6E−03	9.1E−02
F8	−3.8E+3	3.0E+02	−2.6E+03	3.0E+02	−2.5E+03	3.4E+02	−3.2E+03	5.0E+02	−3.4E+03	3.5E+02	−3.2E+03	2.8E+02
F9	9.8	3.9	1.3E+01	4.3	1.6E+01	8.8	2.1E+01	3.5	1.4E+01	9.1	1.2E+01	4.9
F10	7.6E−08	1.8E−07	5.3E−05	1.5E−05	5.7E−02	2.5E−01	6.1E−06	3.1E−06	6.2E−08	6.4E−08	2.8E−01	6.4E−01
F11	8.0E−02	5.2E−02	1.9E−01	8.5E−02	1.9E−01	1.1E−01	2.9E−01	7.7E−02	1.6E−01	1.0E−01	2.7E−01	7.8E−02
F12	1.9E−15	4.0E−15	4.7E−32	4.6E−34	1.4	1.4	1.1E−11	1.6E−11	9.3E−02	2.1E−01	3.1E−02	9.9E−02
F13	1.1E−06	3.4E−06	1.3E−03	2.8E−04	1.1E−03	3.4E−03	1.1E−10	1.8E−10	3.3E−15	4.4E−15	4.7E−03	7.9E−03
F14	9.9E−01	0	1.4	7.0E−01	1.3	4.8E−01	9.9E−01	6.0E−07	1.2	6.2E−01	9.9E−01	1.8E−11
F15	8.8E−04	2.5E−04	3.0E−04	6.2E−10	8.3E−04	1.5E−04	1.0E−03	6.9E−05	8.5E−04	2.8E−04	6.1E−04	1.6E−04
F16	−1.0	0	−1.0	0	−1.0	1.2E−15	−1.0	1.0E−11	−1.0	0	−1.0	2.2E−07
F17	3.9E−01	0	3.98E−01	0	3.98E−01	1.99E−14	3.98E−01	9.60E−12	3.98E−01	0	3.98E−01	3.4E−07
F18	3.0E	2.1E−16	3.0	2.9E−16	3.0	1.0E−13	3.0	7.8E−14	3.0	1.4E−14	3.0	1.0E−06
F19	3.0	1.2E−15	3.00	0	3.0	1.9E−13	3.0	5.3E−14	3.0	1.1E−15	3.0	1.8E−06
F20	−3.8	6.3E−16	−3.8	8.6E−16	−3.8	1.3E−14	−3.8	7.4E−16	−3.8	9.3E−16	−3.8	3.3E−07
F21	−3.2	6.1E−02	−3.2	6.2E−02	−3.3	1.8E−13	−3.3	1.9E−06	−3.2	5.7E−02	−3.3	5.0E−02
F22	−1E+01	0	−5.9	3.7	−8.4	2.9	−1E+01	1.0E−03	−8.1	1.0E−15	−7.8	3.0
F23	−9.6	3.6	−8.3	3.3	−5.8	3.2	−1E+01	5.5E−02	−8.5	3.0	−9.3	2.2

Moreover, the p value from Wilcoxon rank sum test is used as well. Table 5 shows the statistical results obtained for 23 benchmark functions using SA-MFO, PSO, ALO, ABC, MFO, and MVO. As it can be observed from Table 5, SA-MFO outperforms the other algorithms for both unimodal and multimodal test function in most of the functions. The second best results belong the MFO and PSO algorithms. Considering the characteristic of both multi-modal and uni-modal test functions and the obtained results, it may be concluded that SA-MFO algorithm has a high exploration capability for exploring the promising area in the search space and the capability for avoiding the local optima.

For further analyzing the performance of the proposed SA-MFO algorithm, convergence curves for all the used benchmark functions are shown in Fig. 2. In this figure, the performance of the proposed mimetic algorithm is compared with MFO, ABC, MVO, PSO, and ALO. As it can be seen, the proposed SA-MFO algorithm overtakes the other algorithms in most of the cases in terms of both stability and high performance. Moreover, it can be observed that ABC in the most case provides the lowest score. PSO and MVO are in second place. Figure 3 shows the mathematical representation of some benchmark functions, average fitness values calculated for whole moths in each iteration, trajectory changes for the first moth in the first dimension during optimization. As it can be observed, the fluctuations of the first moth are gradually decreased over the course of the iteration. Through this behavior, it can be guaranteed the transition between exploration and exploitation of the search space. Also, Fig. 3 compares the convergence curves of the proposed algorithm with original MFO. From all these figures, it can be observed that using metropolis mechanism can significantly enhance the performance of the original MFO.

From all the obtained results, it can be observed that the proposed SA-MFO in some cases obtained similar results as the original MFO. The explanation for these results is that Metropolis mechanism is only applied in case of the current solution is worse than the best solution obtained so far. Thus, in some cases, based on the probability of acceptance, the current solution is not accepted. Therefore, SA-MFO obtains similar results as original MFO.

A non-parametric statistical test namely Wilcoxon rank sum test is conducted in this paper. This test is used to verify if a pair of two solutions is statistically different or not [40]. In this paper, Wilcoxon rank sum test is used to compare and analyze the difference of the solution (population) of the proposed algorithm with other algorithms. The best values of p when p value < 0.05 which can be considered sufficient evidence against the null hypothesis. The obtained p values produced by Wilcoxon's test of 23 benchmark functions for the pair-wise comparison of the best score obtained for all 1000 iterations with a 5% significance level from the employed statistical test are reported in Table 6. Such pair-wise comparison is constituted by SA-MFO vs. MFO, SA-MFO vs. ALO, SA-MFO vs. PSO, SA-MFO vs. ABC, and SA-MFO vs. MVO. In this table, N/A denotes "not applicable," which mean that the proposed algorithm could not be compared with the other algorithm in the rank-sum test. As it can be observed from this table, SA-MFO outperforms the other algorithms for both unimodal and multimodal test function. As the SA-MFO provides p value less than 0.05 for most of the functions. Also, it can be seen that SA-MFO reject the null hypothesis for F_{16} and F_{17} for most of the competitive algorithms. From all the obtained results, it can be concluded that the SA-MFO algorithm has high exploitation and exploration compared with the original version MFO and other meta-heuristic algorithms. Regarding the p value, it can be concluded that SA-MFO is statistically significant.

Comparison using engineering design problems experiment

In this section, the performance of the proposed SA-MFO is evaluated using four well-known engineering problems, namely three-bar truss, welded beam, pressure vessel and tension/compression spring design problems. The detailed description of the adopted engineering problem is presented in Appendix 2. Table 7 presents the optimal solution for three-bar truss design problem obtained by particle swarm optimization with differential evolution (PSODE) [20], dynamic stochastic selection (DEDS) [42], mine blast algorithm (MBA) [29], water cycle algorithm (WCA) [6], MFO and SA-MFO. As it can be seen from this table, SA-MFO obtained better results compared with the standard MFO. Table 8 compares the optimal solution obtained for pressure vessel problem by SA-MFO with hybrid particle swarm optimization (HPSO) [10], co-evolutionary particle swarm optimization using Gaussian distribution (CPSOGD) [27], GA based on using of dominance-based tour tournament selection (GA-TTS) [3], co-evolutionary differential evolution (CDE) [12], PSO [15], hybrid Nelder-mead simplex search and particle swarm optimization (NMPSO) [41], Gaussian quantum-behaved particle swarm optimization (GQPSO) [1], and MFO. As it can be observed, SA-PSO and ACO obtained the optimal solutions compared with the other algorithms.

Table 9 compares the best obtained results for tension/compression spring design problem of SA-MFO and CPSO, DEDS, NM-PSO, GA-TTS, hybrid evolutionary algorithm and adaptive constraint handling technique (HEAA) [39], mine blast algorithm (MBA) [30], WCA, differential evolution with level comparison (DELC) [38], and MFO. In this table, NA indicates "not available" value. As it can be seen, NM-PSO obtained the optimal solution. However, it can be observed that SA-MFO obtained better results compared with original MFO. The best solutions for welded beam prob-

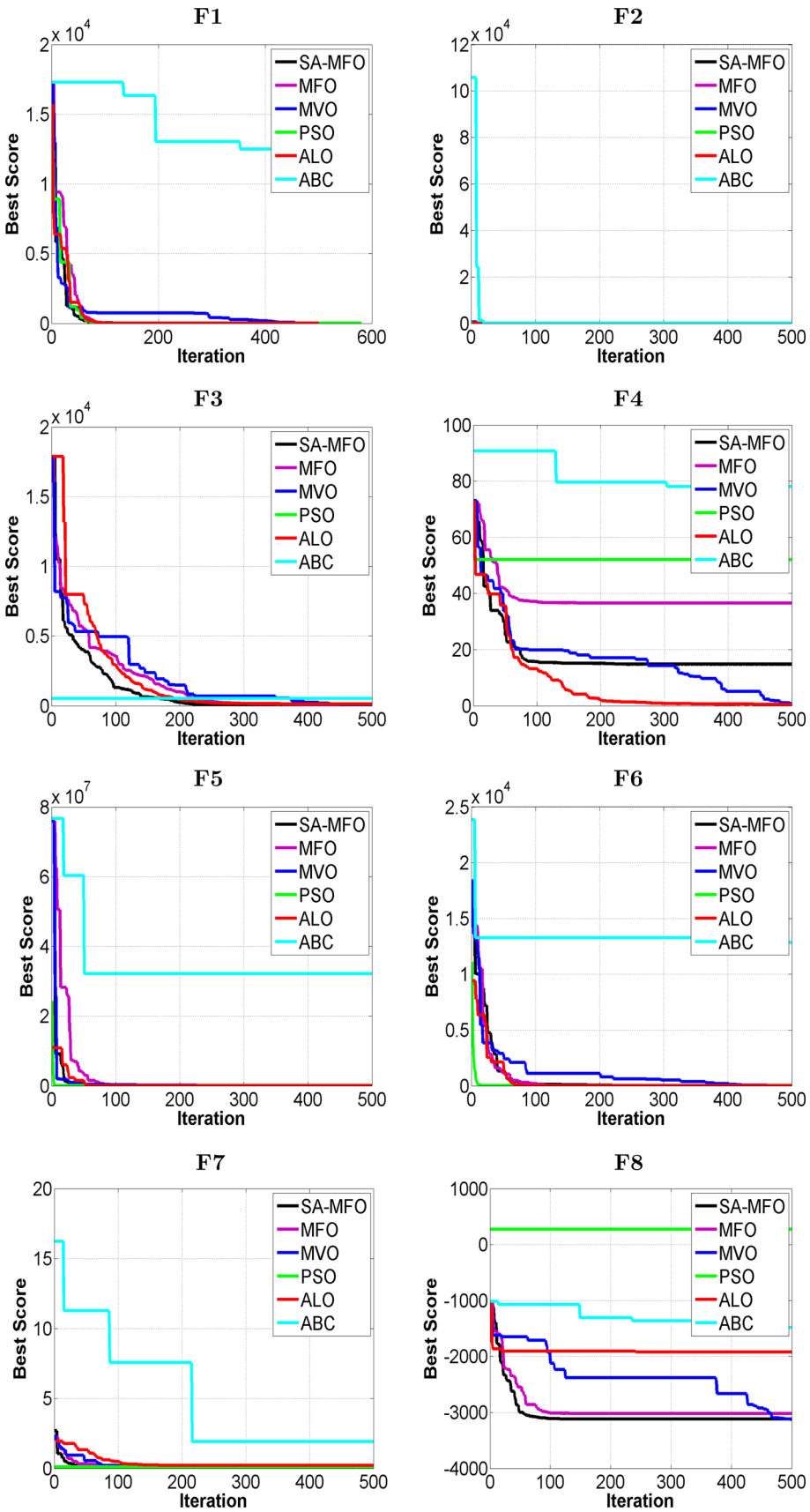


Fig. 2 Convergence curves of F1 to F23

Fig. 2 continued

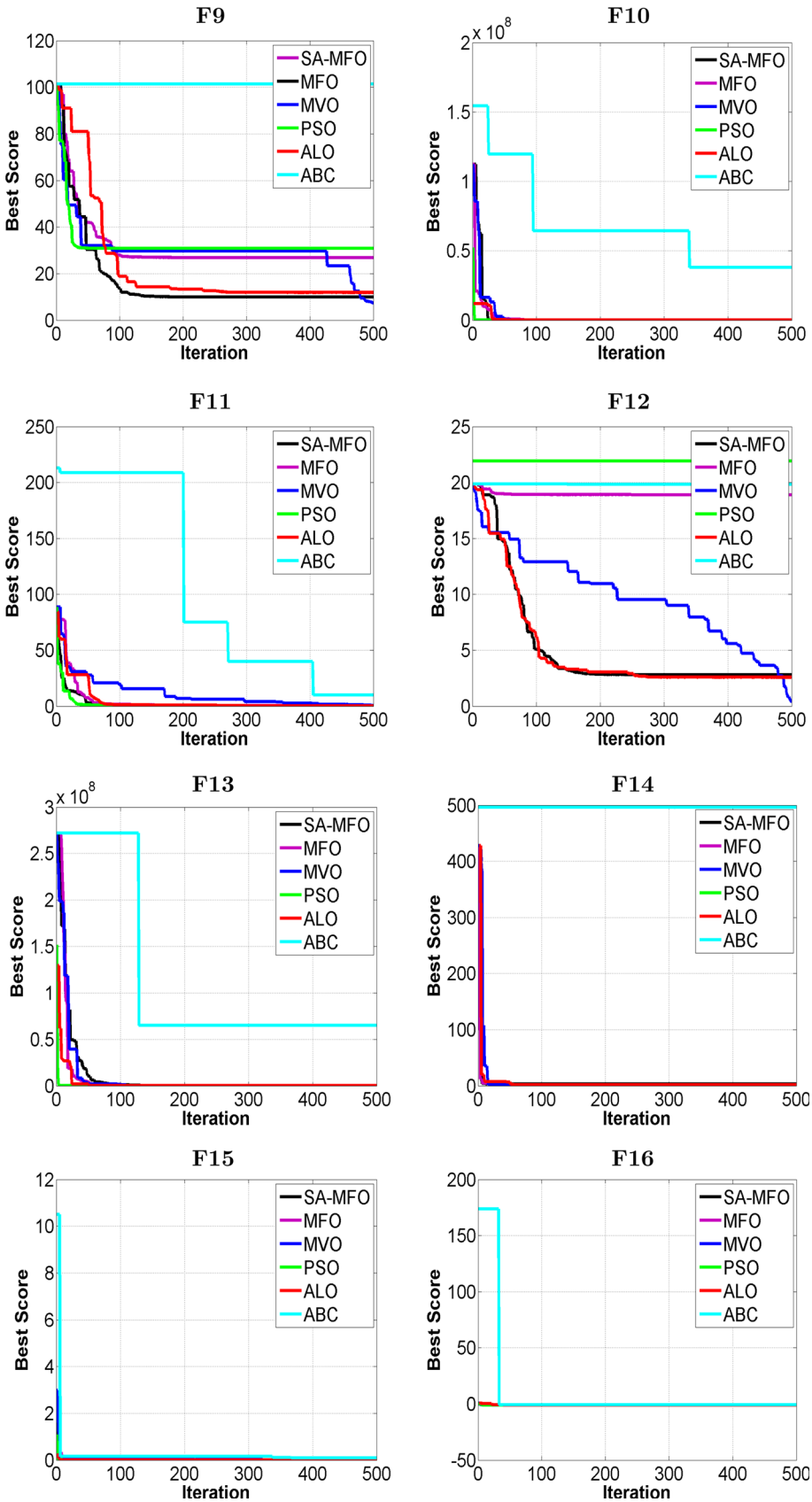
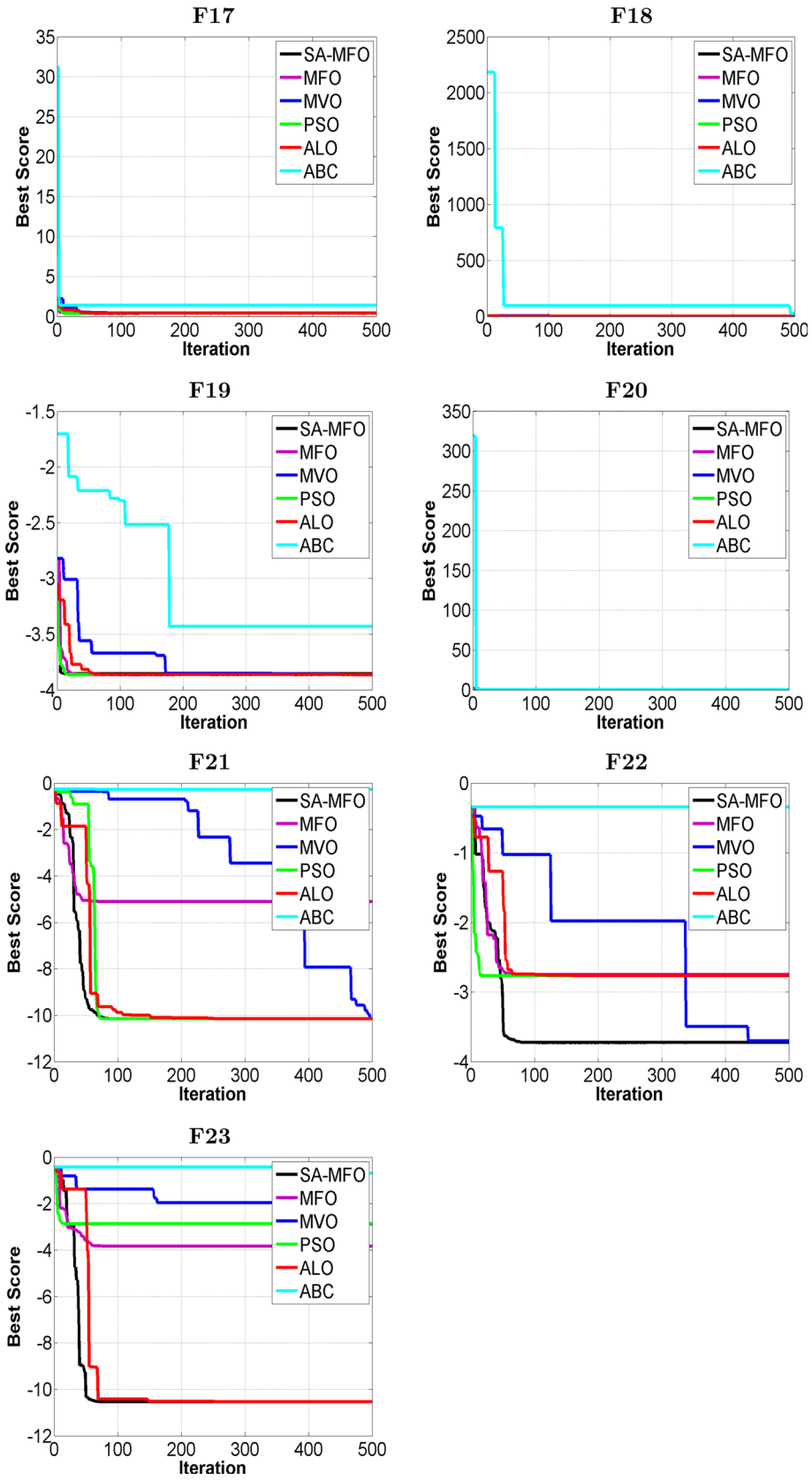


Fig. 2 continued



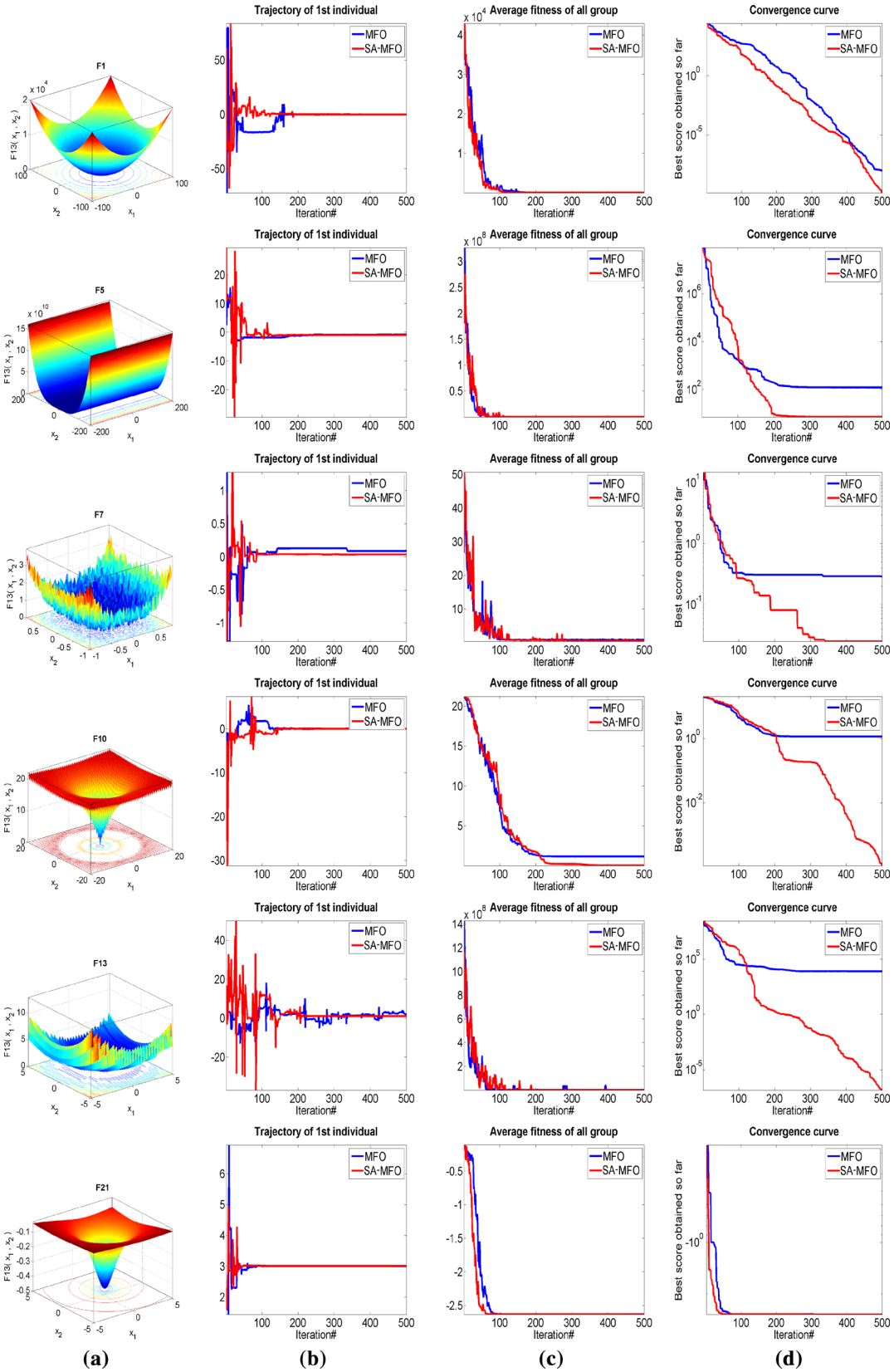


Fig. 3 **a** Graphical representations of benchmark functions, **b** the trajectory in first dimension, **c** the average fitness of all group, and **d** the convergence curve

Table 6 Statistical and p values of the Wilcoxon ranksum test results for 23 benchmark functions

	SA-MFO vs. PSO	SA-MFO vs. ALO	SA-MFO vs. ABC	SA-MFO vs. MFO	SA-MFO vs. MVO
F1	3.02E−11	2.86E−09	2.45E−11	0.207275	1.06E−11
F2	1.94E−07	3.94E−11	3.02E−11	5.49E−06	1.02E−13
F3	2.58E−07	1.86E−09	5.57E−10	0.000491	0.491783
F4	3.02E−05	5.46E−03	4.31E−08	5.21E−01	1.11E−06
F5	3.16E−05	0.042896	0.053685	4.38E−06	1.123E−06
F6	8.01E−05	6.80E−08	7.95E−07	7.15E−05	6.80E−08
F7	1.66E−07	0.008355	0.101729	0.675014	8.29E−05
F8	5.96E−07	4.49E−07	6.72E−08	0.000273	0.001619
F9	0.000148	1.64E−06	2.69E−06	8.50E−05	0.007764
F10	2.86E−08	6.80E−08	6.80E−08	0.092457	6.80E−08
F11	0.001315	0.009097	0.017257	0.003447	0.005390
F12	0.000111	0.000183	0.000183	5.71E−03	0.000183
F13	6.39E−05	6.39E−05	0.002202	0.062318	0.001706
F14	0.034843	5.72E−05	6.39E−05	0.000368	6.39E−05
F15	0.000183	0.733730	0.025748	0.820530	0.001706
F16	N/A	6.29E−05	6.39E−05	N/A	6.39E−05
F17	N/A	6.34E−05	6.39E−05	N/A	6.39E−05
F18	0.025395	0.000149	0.000149	0.005788	0.000149
F19	0.005745	0.000165	0.000165	0.937947	0.000165
F20	0.016749	6.39E−05	0.013652	0.012345	6.39E−05
F21	0.550533	0.467611	0.467611	0.930111	9.44E−01
F22	0.000729	6.39E−05	6.39E−05	0.000767	6.39E−05
F23	0.006848	0.001707	0.135064	0.015354	0.013506

Table 7 Comparison of best obtained results using various optimization algorithms for three-bar truss design problem

	DEDS	PSO-DE	WCA	MBA	MFO	SA-MFO
z_1	0.788675	0.788675	0.788651	0.788565	0.78834477	0.78839302
z_2	0.408248	0.408248	0.408316	0.4085597	0.40946691	0.40904779
$h_1(\mathbf{z})$	1.77E−08	−5.29E−11	0	−5.29E−11	−1.00E−09	−1.7E−11
$h_2(\mathbf{z})$	−1.464101	−1.463747	−1.464024	−1.4637475	−1.46440134	−1.46319344
$h_3(\mathbf{z})$	−0.535898	−0.536252	−0.535975	−0.5362524	−0.53559870	−0.53680729
$f(\mathbf{z})$	263.895843	263.895843	263.895843	263.8958522	263.89597968	263.89595986

lem obtained by WCA, GA-TTS, cultural algorithms with evolutionary programming (CAEP) [2], MFO, HGA, MBA, CPSO, NM-PSO, gravitational search algorithm (GSA) [26], HPSO, and SA-MFO are shown in Table 10. From this table, SA-MFO overtakes the other algorithms.

Computational complexity of the proposed SA-MFO algorithm

In this section, the computational complexity of the proposed hybrid algorithm is computed. The computational complexity of a meta-heuristic algorithm is mainly depended on the number of variables, the number of moths, the flames' sorting mechanism in each iteration and the maximum number of iterations [24]. Since the original MFO uses Quicksort mechanism, the computational complexity is $O(n^2)$ in the worst case. In addition, the temperature in simulated annealing goes through $O(log(m))$ [9]. The overall computational complexity is $O(T \times log(m)(m^2 + m \times v))$, where T is the maximum number of iterations and v is the number of variables.

Conclusions

This paper introduced a new hybrid algorithm based on using MFO and SA. The metropolis acceptance mechanism of SA is employed into the standard MFO to enhance the performance of MFO. The simulation results using 23 benchmark functions show that the proposed SA-MFO is superior compared with the original version MFO and other

Table 8 Comparison of best obtained results using various optimization algorithms for pressure vessel problem

	z_1	z_2	z_3	z_4	$h_1(\mathbf{z})$	$h_2(\mathbf{z})$	$h_3(\mathbf{z})$	$h_4(\mathbf{z})$	$f(\mathbf{z})$
GA-TTS	0.8125	0.4375	42.0974	176.654	−2.01E−03	−3.58E−02	−24.7593	−63.346	6059.9463
CDE	0.8125	0.4375	42.0984	176.6376	−6.67E−07	−3.58E−02	−3.705123	−63.3623	6059.734
CPSO	0.8125	0.4375	42.0913	176.765	−1.37E−06	−3.59E−04	−118.7687	−63.2535	6061.0777
HPSO	0.8125	0.4375	42.0984	176.6366	−8.80E−07	−3.58E−02	3.1226	−63.3634	6059.7143
NM-PSO	0.8036	0.3972	41.6392	182.412	3.65E−05	3.79E−05	−1.5914	−57.5879	5930.3137
G-QPSO	0.8125	0.4375	42.0984	176.6372	−8.79E−07	−3.58E−02	−0.2179	−63.3628	6059.7208
ACO	0.8125	0.4375	42.1036	176.5726	NA	NA	NA	NA	6059.0888
GSA	1.125	0.625	55.9886	84.4542	NA	NA	NA	NA	8538.8359
MFO	0.8125	0.4375	43.7297397	167.6570310	−0.0016659	−0.0048997	−828.5755107	−82.4429690	6059.8413000
SA-MFO	0.8125	0.4375	42.1035	167.56230	−0.0000398	−0.0001184	−1517.6393667	−69.0752	6059.0888

Table 9 Comparison of best obtained results using various optimization algorithms for tension/compression spring design problem

	DEDS	GA-TTS	CPSO	HEAA	NM-PSO	DELC	MBA	WCA	MFO	SA-MFO
z_1	0.051689	0.051989	0.051728	0.051689	0.05162	0.051689	0.051656	0.05168	0.052444	0.050000
z_2	0.356717	0.363965	0.357644	0.356729	0.355498	0.356717	0.35594	0.356522	0.375164	0.317402
z_3	11.288965	10.890522	11.244543	11.288293	11.333272	11.288965	11.344665	11.30041	10.284374	14.030933
$h_1(\mathbf{z})$	1.45E−09	−1.26E−03	−8.25E−04	3.96E−10	1.01E−03	−3.40E−09	0.00	1.65E−13	−1.97E−05	−2.91E−06
$h_2(\mathbf{z})$	−1.19E−09	−2.54E−05	−2.52E−05	3.59E−10	9.94E−04	2.44E−09	0.00	−7.90E−14	−3.71E−09	−5.84E−05
$h_3(\mathbf{z})$	−4.053785	−4.061337	−4.051306	−4.053808	−4.061859	−4.053785	−4.052248	−4.053399	−4.088642	−3.968053
$h_4(\mathbf{z})$	−0.727728	−0.722697	−0.727085	−0.72772	−0.728588	−0.727728	−0.728268	−0.727864	−0.714928	−0.755065
$f(\mathbf{z})$	0.012665	0.012681	0.012674	0.012665	0.01263	0.012665	0.012665	0.012665	0.012676	0.0126721

Table 10 Comparison of best obtained results using various optimization algorithms for welded beam problem

	GA-TTS	CPSO	CAEP	HGA	NM-PSO	WCA	GSA	MBA	HPSO	MFO	SA-MFO
$z_1(h)$	0.20599	0.202369	0.2057	0.2057	0.20583	0.205728	0.1821	0.20573	0.20573	0.2051	0.2057
$z_2(l)$	3.47124	3.544214	3.4705	3.4705	3.468338	3.470522	0.8570	3.47049	3.470489	3.4847	3.4724
$z_3(t)$	9.02022	9.04821	9.0366	9.0366	9.036624	9.03662	10	9.03663	9.036624	9.0365	9.0367
$z_4(b)$	0.2065	0.205723	0.2057	0.2057	0.20573	0.205729	0.2024	0.20573	0.20573	0.2057	0.2057
$h_1(\mathbf{z})$	− 0.10305	− 13.65555	1.988676	1.988676	− 0.02525	− 0.034128	NA	− 0.00161	− 0.025399	− 5.4387	− 1.2051
$h_2(\mathbf{z})$	− 0.23175	− 78.81408	4.481548	4.481548	− 0.053122	− 3.49E−05	NA	− 0.01691	− 0.053122	− 13.0669	− 0.5520
$h_3(\mathbf{z})$	− 5.0E−04	− 3.35E−03	0	0	0.0001	− 1.19E−06	NA	− 2.4E−07	0	− 0.0006	− 0.0001
$h_4(\mathbf{z})$	− 3.43044	− 3.424572	− 3.433213	− 3.433213	− 3.433169	− 3.43298	NA	− 3.43298	− 3.432981	− 3.4314	− 3.4328
$h_5(\mathbf{z})$	− 0.08099	− 0.077369	− 0.0507	− 0.0807	− 0.08083	− 0.080728	NA	− 0.08073	− 0.08073	− 0.0801	− 0.0807
$h_6(\mathbf{z})$	− 0.23551	− 0.235595	− 0.235538	− 0.235538	− 0.23554	− 0.23554	NA	− 0.23554	− 0.23554	− 0.2355	− 0.2355
$h_7(\mathbf{z})$	− 58.6469	− 4.472858	2.603347	2.603347	− 0.031555	− 0.013503	NA	− 0.00146	− 0.031555	− 1.3454	− 0.0626
$f(\mathbf{z})$	1.72823	1.728024	1.724852	1.724852	1.724717	1.724856	1.880	1.72483	1.724852	1.7249	1.7245

recently meta-heuristic algorithms, namely GWO, ALO, PSO, and MVO. Moreover, the paper considered solving four well-known engineering design problems using SA-MFO. The experimental results show that the proposed SA-MFO obtained competitive results compared with the other algorithms proposed in the literature for solving engineering problems. Future work could concentrate on applying the proposed SA-MFO on more complex optimization problems.

Appendix 1: List of 23 benchmark optimization functions

See Tables 11 and 12.

Table 11 Benchmark functions

No.	Definition
$F1$	$f_1(x) = \sum_{i=1}^{n} x^{i^2}$
$F2$	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
$F3$	$f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$
$F4$	$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$
$F5$	$f_5(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
$F6$	$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$
$F7$	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + U(0, 1)$
$F8$	$f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$
$F9$	$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10n]$
$F10$	$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + \epsilon$
$F11$	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
$F12$	$f_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_i + 1)] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\, y_i = 1 + \frac{x_i+1}{4}\, u(x_i, 10, 100, 4) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$
$F13$	$f_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
$F14$	$f_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1}$
$F15$	$f_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$
$F16$	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
$F17$	$f_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$

Table 11 continued

No.	Definition
$F18$	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 - 3x_2)^2 * (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
$F19$	$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$
$F20$	$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$
$F21$	$f_{21}(x) = -\sum_{i=1}^{5} [(x - a_i)(x - a_i)^T + c_i]^{-1}$
$F22$	$f_{22}(x) = -\sum_{i=1}^{7} [(x - a_i)(x - a_i)^T + c_i]^{-1}$
$F23$	$f_{23}(x) = -\sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$

Table 12 Properties of benchmark functions, lb denotes lower bound, ub denotes upper bound, dim denotes dimensions, opt denotes optimum point

No.	lb	ub	opt	dim	Modality
$F1$	-100	100	0	50	Unimodal
$F2$	-10	10	0	50	Unimodal
$F3$	-100	100	0	50	Unimodal
$F4$	-100	100	0	50	Unimodal
$F5$	-30	30	0	50	Unimodal
$F6$	-100	100	0	50	Unimodal
$F7$	-1.28	1.28	0	50	Unimodal
$F8$	-500	500	-418.9829×50	50	Multimodal
$F9$	-5.12	5.12	0	50	Multimodal
$F10$	-32	32	0	50	Multimodal
$F11$	-600	600	0	50	Multimodal
$F12$	-50	50	0	50	Multimodal
$F13$	-50	50	0	50	Multimodal
$F14$	-65	65	1	2	Multimodal
$F15$	-5	5	0	4	Multimodal
$F16$	-5	5	-1.0316	2	Multimodal
$F17$	-5	5	0.398	2	Multimodal
$F18$	-2	2	3	2	Multimodal
$F19$	1	3	-3.86	3	Multimodal
$F20$	0	1	-3.32	6	Multimodal
$F21$	0	10	-10.1532	4	Multimodal
$F22$	0	10	-10.4028	4	Multimodal
$F23$	0	10	-10.5363	4	Multimodal

Appendix 2. Details of the 4 constrained design benchmark optimization problems

Compression spring design problem

This is a minimization problem which contain three main variables are coil diameter (CD), wire diameter (WD), and the number of active coils (NAC) under some restrictions such as minimum deflection, surge frequency, and shear stress. This problem can be mathematically formulated as follows:

Minimize the function (7)
$f(\mathbf{z}) = (z_3 + 2)z_2z_1^2$, $\mathbf{z} = [z_1z_2z_3] = (\text{CD})(\text{WD})(\text{NAC})$,

Subject to,
$h_1(\mathbf{z}) = 1 - \frac{z_2^3 z_3}{717854 z_1^4} \leq 0$,
$h_2(\mathbf{z}) = \frac{4z_2^2 - z_1z_2}{12566(z_2z_1^3 - z_1^4)} + \frac{1}{5108z_1^2} - 1 \leq 0$,
$h_3(\mathbf{z}) = 1 - \frac{140.45z_1}{z_2^2 z_3} \leq 0$,
$h_4(\mathbf{z}) = \frac{z_1 + z_2}{1.5} - 1 \leq 0$

where,
$0.05 \leq z_1 \leq 2$,
$0.25 \leq z_2 \leq 1.30$,
$2 \leq z_3 \leq 15$

The design of welded beam problem

Welded beam design is a minimization problem of the cost under some constraints where there are four variables in this problem which are the length of bar attached to the weld (l), weld thickness (h), the height of the bar (t), and the thickness of the bar (b). Moreover, there are some constraints in this problem which are bending stress in the beam (α), the beam deflection (β), buckling load on the bar (BL), the end deflection of the beam (δ), and side constraints. This design problem can then formulated as follows:

Minimize the function (8)
$f(\mathbf{z}) = 1.10471z_1^2z_2 + 0.04811z_3z_4(14.0 + z_2)$,
$\mathbf{z} = [z_1z_2z_3] = [hltb]$,

Subject to,
$h_1(\mathbf{z}) = \delta(\mathbf{z}) - \delta_{\max} \leq 0$,
$h_2(\mathbf{z}) = \alpha(\mathbf{z}) - \alpha_{\max} \leq 0$,
$h_3(\mathbf{z}) = z_1 - z_4 \leq 0$,
$h_4(\mathbf{z}) = 0.10471z_1^2 + 0.04811z_3z_4(14.0 + z_2) - 5.0 \leq 0$,
$h_5(\mathbf{z}) = 0.125 - z_1 \leq 0$,
$h_6(\mathbf{z}) = \delta(\mathbf{z}) - \delta_{\max} \leq 0$,
$h_7(\mathbf{z}) = B - BL(\mathbf{z}) \leq 0$

where,
$0.1 \leq z_1 \leq 2$,
$0.1 \leq z_2 \leq 10$,
$0.1 \leq z_3 \leq 10$,

$0.1 \leq z_4 \leq 20$
and
$\delta(\mathbf{z}) = \sqrt{\delta'^2 + 2\delta'\delta^n \frac{z_2}{2R} + \delta^{n2}}$
$\delta' = \frac{B}{\sqrt{2}z_1 z_2}$
$\delta^n = \frac{MR}{J}$
$M = B(L + \frac{z_2}{2})$
$R = \sqrt{\frac{z_2^2}{4} + (\frac{z_1+z_3}{2})^2}$
$J = 2(\sqrt{2}z_1 z_2[\frac{z_2^2}{12} + (\frac{z_1+z_3}{2})^2])$
$\alpha(\mathbf{z}) = \frac{6PL}{z_4 z_3^2}$, $\beta(\mathbf{z}) = \frac{4PL^3}{Ez_3^2 z + z_4}$
$BL(\mathbf{z}) = \frac{4.013E\sqrt{\frac{z_3^2 z_4^6}{36}}}{L^2}(1 - \frac{z_3}{2L}\sqrt{\frac{E}{4G}})$
$P = 6000lb,\ L = 14in.,\ \beta_{\max} = 0,\ 25in.,$
$E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\delta_{\max} =$ 136,00 psi, $\alpha_{\max} = 30,000$ psi

Design of pressure vessel problem

The main objective of the problem of pressure vessel design is to minimize the total cost of the welding and forming of pressure vessel problem. In this problem there are four variables which are thickness of the head (T_h), thickness of the shell (T_s), inner radius (R), and length of the cylindrical section of the vessel (L).

Minimize the function (9)
$f(\mathbf{z}) = 0.6224z_1 z_2 z_3 z_4 + 1.7781 z_2 z_3^2 + 3.1661 z_1^2 z_4$
$+ 19.84 z_1^2 z_3$, $(\mathbf{z}) = [z_1 z_2 z_3] = [T_s T_h R L]$,

Subject to,
$h_1(\mathbf{z}) = -z_1 + 0.0193 z_3 \leq 0$,
$h_2(\mathbf{z}) = -z_3 + 0.00954 z_3 \leq 0$,
$h_3(\mathbf{z}) = -z_3^2 z_4 - 4/3 z_3^3 \leq 0$,
$h_4(\mathbf{z}) = z_4 - 240 \leq 0$,

where,
$1.1 \leq z_1 \leq 12.5$,
$0.6 \leq z_2 \leq 12.5$,
$0 \leq z_3 \leq 240$,
$0 \leq z_4 \leq 240$

The three-bar trust design problem

This is a minimization problem. This problem can be described mathematically as follows:

Minimize the function
$f(\mathbf{z}) = (2\sqrt{2}z_1 + z_2) \times l$,

Subject to,
$h_1(\mathbf{z}) = \frac{\sqrt{2}z_1 + z_2}{\sqrt{2}z_1^2 + 2z_1 z_2} p - \alpha \leq 0$,
$h_2(\mathbf{z}) = \frac{z_2}{\sqrt{2}z_1^2 + 2z_1 z_2} p - \alpha \leq 0$,
$h_3(\mathbf{z}) = \frac{1}{\sqrt{2}z_2 + z_1} p - \alpha \leq 0$,

where,
$0 \leq z_i \leq 1, i = 1, 2$,
$l = 100$ cm, $p = 2$ kN/cm^2 $\alpha = 2$ kN/cm^2.
(10)

References

1. Coelho L (2010) Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Syst Appl 37:1676–1683
2. Coello C, Becerra R (2004) Efficient evolutionary optimization through the use of a cultural algorithm. Eng Optim 36:219–236
3. Coello C, Mezura E (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 16:193–203
4. Colorni A, Dorigo M, Maniezzo V (1992) Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life. Paris, France, p 134–142
5. Emary E, Zawbaa H (2016) Impact of chaos functions on modern swarm optimizers. PLoS One 11(7):1–26
6. Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110:151–166
7. Gaber T, Sayed G, Anter A, Soliman M, Ali M, Semary N et al (2015) Thermogram breast cancer prediction approach based on neutrosophic sets and fuzzy c-means algorithm. In: The 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC). Milan, Italy, pp 4254–4257
8. Gong W, Cai Z, Ling C (2010) De/bbo: a hybrid differential evolution with biogeography-based optimization for global numerical optimization. Soft Comput 15(4):645–665
9. Hansen PB (1992) Simulated annealing. Electr Eng Comput Sci Tech Rep 170:1–14
10. He Q, Wang L (2007) A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl Math Comput 186:1407–1422
11. Henderson D, Jacobson S, Johnson A (2003) The theory and practice of simulated annealing. Handb Metaheuristics 57:287–319
12. Huang F, Wang L, He Q (2007) An effective co-evolutionary differential evolution for constrained optimization. Appl Math Comput 186(1):340–356
13. Kao T, Zahara E (2008) A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Appl Soft Comput 8(2):849–857
14. Karaboga D, Basturk B (2007) Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In: 12th international fuzzy systems association world congress, vol 4529. Mexico, pp 789–798

15. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE international conference on neural networks. Perth, WA, pp 1942–1948
16. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
17. Li Z, Zhou Y, Zhang S, Song J (2016) Lévy-flight moth-flame algorithm for function optimization and engineering design problems. Math Probl Eng 2016:1–22
18. Liang J, Suganthan P, Deb K (2005) Novel composition test functions for numerical global optimization. In: Proceedings 2005 IEEE swarm intelligence symposium. p 68–75
19. Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10(2):629–640
20. Liu H, Cai Z, W Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10:629–640
21. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH (1953) Equation of state calculations by fast computer machines. J Chem Phys 21(6):10871092
22. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
23. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
24. Mirjalili SM (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst (Elsevier) 89:228–249
25. Mohamed A (2017) Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm. Complex Intell Syst 3(4):205–231
26. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) Gsa: a gravitational search algorithm. Inf Sci 179:2232–2248
27. Renato A, Leandro D, Santos C (2006) Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. IEEE Trans Syst Man Cybern Part C (Applications and Reviews) 36:1407–1416
28. Rizk-Allah R, Hassanien A (2017) New binary bat algorithm for solving 0–1 knapsack problem. Complex Intell Syst. https://doi.org/10.1007/s40747-017-0050-z
29. Sadollaha A, Bahreininejada A, Eskandarb H, Hamdi M (2013) Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13(5):2592–2612
30. Satapathy S, Naik A (2016) Social group optimization (sgo): a new population evolutionary optimization technique. Complex Intell Syst 2(3):173–203
31. Sayed G, Darwish A, Hassanien A (2017) Quantum multiverse optimization algorithm for optimization problems. Neural Comput Appl. https://doi.org/10.1007/s00521-017-3228-9
32. Sayed G, Hassanien A (2017) Moth-flame swarm optimization with neutrosophic sets for automatic mitosis detection in breast cancer histology images. Appl Intell 47(2):397408
33. Sayed G, Hassanien A, Azar A (2017) Feature selection via a novel chaotic crow search algorithm. Neural Comput Appl. https://doi.org/10.1007/s00521-017-2988-6
34. Shieh H, Kuo C, Chiang C (2011) Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification. Appl Math Comput 218(8):4365–4383
35. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
36. Sun J, Sun F, Gong D, Zeng X (2017) A mutation operator guided by preferred regions for set-based many-objective evolutionary optimization. Complex Intell Syst 3(4):265–278
37. Wang CY, Lin M, Zhong YW, Zhang H (2016) Swarm simulated annealing algorithm with knowledge-based sampling for travelling salesman problem. Int J Intell Syst Technol Appl 15(1):74–94
38. Wang L, Li L (2010) An effective differential evolution with level comparison for constrained engineering design. Struct Multidiscip Optim 41:947–963
39. Wang Z, Cai Y, Zhou Y, Fan Z (2009) Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint handling technique. Struct Multidiscip Optim 37:395–413
40. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1:80–83
41. Zahara E, Kao Y (2009) Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Syst Appl 36:3880–3886
42. Zhang M, Luo W, Wang X (2008) Differential evolution with dynamic stochastic selection for constrained optimization. Inf Sci 178:3043–3074