Journal of Structural Engineering & Applied Mechanics (2022) 5(2):96-116
DOI 10.31462/jseam.2022.02096116

RESEARCH ARTICLE

Solution of design optimization problems via metaheuristic search methods

Betül Üstüner, Erkan Doğan

Manisa Celal Bayar University, Department of Civil Engineering, Manisa, Türkiye

Article History

Received 24 January 2022
Accepted 11 June 2022

Keywords

Swarm intelligence
Refined firefly algorithm
Benchmark problem
Optimum design

Abstract

Metaheuristic algorithms inspired by natural phenomena are frequently used in solving optimization problems recently. Just as every problem has its characteristics, every algorithm has its unique structure. Therefore, problem-specific algorithm selection is an important issue. In addition, metaheuristic algorithms are very open to development. Therefore, improved/modified versions of algorithms are common. Working with benchmarking problems and engineering design problems is the best way to compare the performance and reliability of metaheuristic algorithms. In this study, the performances of firefly (FA), particle swarm optimization (PSO), bat algorithm (BA), ant colony optimization (ACO), glow worms (GSO), and hunting search (HuS) algorithms are compared. An enhanced version of the firefly algorithm (RFA) is also recommended and included in the comparison. A benchmark and five engineering design problems were selected for comparison purposes. All algorithms are set to twenty thousand iterations. The results show that the metaheuristic algorithm that gives the best results varies according to the nature of the problem. Moreover, although it does not change the ranking of the algorithm that gives the best result according to the problem, it shows that RFA gives better results in every problem than FA.

1. Introduction

The method of searching for the best solution to the problem under certain conditions is called optimization. Optimization solution methods can generally be examined under two headings as mathematical and heuristic optimization algorithms [1]. Mathematical algorithms scan the entire solution space, so the computational load of these algorithms is higher. Therefore, in complex problems, it is more advantageous to use heuristic algorithms that solve in a shorter time by intuitively scanning the solution space. Today, the successful use of basic heuristics has formed the basis for the development of metaheuristic algorithms. These algorithms are widely used because of their simplicity, flexibility, non-differential mechanisms, and avoidance of local optima [2]. Some of the most popular metaheuristic algorithms are particle swarm optimization (PSO) [3–7], ant colony optimization (ACO) [8–12], bat (BA) [13–18], firefly optimization (FA) [7, 19–22], glow worms (GSO) [23, 24] and hunting search (HuS) [25–28] algorithms. Metaheuristic algorithms are used in the data processing environment because they use more efficient search processes and can reach the best solution faster [29, 30]. These optimization techniques are applied in various fields of study, such as truss design [31, 32], parameter estimation [31, 32], and traffic engineering [33, 34]. Meta-heuristics are simple

Correspondence Betül Üstüner betul.celik@cbu.edu.tr

eISSN 2630-5763

to implement and are inspired by nature and imitated the behavior of animals. By simulating these behaviors, researchers propose new meta-heuristic methods day by day. Meta-heuristic methods can be applied to different problems without the need for much change in algorithm operation. In the software of codes, the input and output values of a system are very important in meta-heuristics. Therefore, the most important step required to solve problems with metaheuristic methods is to determine how to define the problem. While metaheuristic algorithms converge to the solution, they operate a predictive process starting from a random solution and do not need derivative information. Therefore, they differ from mathematical methods in these aspects. In addition, the local minimization drawbacks of traditional optimization methods are overcome by metaheuristic methods that have global search capability since they operate a predictive process [2]. Newly developed metaheuristic algorithms are often tested with benchmark and engineering functions and then compared with other algorithms to confirm their performance [35–37]. These functions are optimized with the most appropriate parameter value to obtain the best solution. All optimization algorithms aim to find the best solution as fast as possible. Test functions can be divided into categories such as modality, basins, valleys and separability, dimensionality. These categories are determined by the features that shape the appearance of the problem [38]. Traditional mathematical gradient-based programming techniques are effective to a certain extent. However, they are not suitable for application in complex design problems, so they are insufficient to reach solutions for some problems. Therefore, there is a need for solution methods that are more successful in global search than gradient-based programming techniques and that use probabilistic processes to obtain optimum designs [39].

In an optimization problem, the objective must be well defined so that the algorithm achieves the best result. The objective function indicates a quantity to be optimized, such as benefit or cost. Then the constraints and design variables must be defined separately. The type of design variables might change depending on the type of problem or need. Constraints are generally rules-dependent variables or system constraints. A general optimization problem can be described as the selection of the best values or variables where the constraints are provided [40]. In general terms, an optimization problem is defined as follows [41, 42].

The objective function is

$$z = F(x) \tag{1}$$

Constraints are

$$g_k(x) \leq 0 \quad k = 1,2,\dots,p \tag{2}$$

$$h_j(x) = 0 \quad j = 1,2,\dots,m \tag{3}$$

Design variable ranges are

$$L_{xk} \leq x \leq U_{xk}, \; k = 1,2,\dots,n \tag{4}$$

The *x* values of design variables that make the objective function maximum or minimum under specific conditions are called optimization problems. F(x) is the objective function. *x* is the design variable, *m* is the equality constraint, and *p* is the inequality constraint.

In this study, the performances of FA, RFA, PSO, BA, ACO, GSO, and HuS algorithms for different problems are compared. In addition, optimization design problems are solved by using the basic version of the firefly algorithm. According to the results, the simple version of the algorithm is stuck in the local optimum point. To overcome this, it is aimed to improve by adding a new term to the search position update formulation. As a result of this improvement, the analysis of the same problems is examined, and it is observed that the problem of getting stuck in the local optimum was overcome.

The second section of the study gives information about the used metaheuristic algorithms. Then, the definitions and solutions of the problems discussed in the third section are explained. In addition, the result tables and the performance graphics of the algorithms are also included in this section. Finally, the last section is the conclusion.

2. Methodology

In this part of the study, FA, RFA, PSO, ACO, BA, GSO, and HuS algorithms used as metaheuristic algorithms in the study are introduced.

2.1. Firefly algorithm

FA was developed by Xin-She Yang in 2008 [43]. This algorithm is designed according to the brightness and movement directions of fireflies. Fireflies, which have 2000 different species, live in tropical and temperate regions [19]. The light emitted by fireflies, which creates very pleasant images, has two basic functions. These functions are the ability to catch their prey by attracting them and the ability to mate by courting their mates [18]. The reason fireflies emit light is photogenic cells on their body surfaces. There is no gender discrimination in fireflies, they are unisex. The lower luminous fireflies move towards the higher luminous fireflies. In the FA, the light intensity I of the firefly is proportional to the value of the fitness function.

$$I(r) = I_0 e^{-\gamma r^2} \tag{5}$$

where I_0 indicates the light intensity of the source and the light absorption is approximated using the constant light absorption coefficient. The allurement of fireflies is proportional to their light intensities $I(r)$.

$$\beta = \beta_0 e^{-\gamma r^2} \tag{6}$$

where β is the allurement at $r = 0$. The steps of the algorithm are given below:

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}\left(x_i - x_j\right) + \alpha\varepsilon_i \tag{7}$$

Here, $\beta_0 e^{-\gamma r^2}\left(x_i - x_j\right)$ is the randomization parameter resulting from the attraction of the firefly x_j and $\alpha\varepsilon_i$. The allurement of the new firefly position is compared with the allurement of the old position. If the new location generates a higher allurement value, the firefly will move to the new location, otherwise, the firefly will stay in its current position. As the termination criterion, FA uses a predefined random number of iterations or a fitness value. The brightest firefly moves randomly according to Eq (8).

$$x_i(t+1) = x_i(t) + \alpha\varepsilon_i \tag{8}$$

2.2. Refined firefly algorithm

Firefly algorithm optimization is one of the heuristic methods based on the swarm technique. The movements of fireflies depend on certain parameters. The fireflies' positions are updated using Eq. (7). In the improved firefly algorithm [44] thanks to the added parameter, the updated positions are added in a way that allows the fireflies to move randomly in certain directions in the near environment of their current position.

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}\left(x_i - x_j\right) + \alpha\varepsilon_i + h_i r_1 \frac{N_s}{\Delta t} \tag{9}$$

$$h_i = \begin{cases} 1 & if \;\; r_2 \leq \dfrac{1}{2N_d} \\ 0 & if \;\; r_2 > \dfrac{1}{2N_d} \end{cases} \tag{10}$$

where, r_1 is a random number between 0 and 1; N_s is referred to as the maximum value of the design variables. $h_i$is 0-1 weight function applied by sampling a random number r_2 between 0 and 1. N_d is the number of dimensions of each firefly. Eq. (9) indicates that the inserted term design variable is allowed to change its position to a new one. This enables fireflies to continue their search for the optimum. Thanks to the reformulated equation, it was observed that the technique increased its effectiveness. Advances in the technique are illustrated with solved numerical examples.

2.3. Particle swarm optimization

PSO [45] was introduced in 1995 by social psychologists Kennedy and Eberhart, an electrical engineer. PSO is a solution algorithm for online, non-convex, or combinatorial optimization problems arising in many fields of science and engineering. The emergence of this algorithm is inspired by a herd of birds or fish. A herd is formed by the gathering of many bird species. The size of the herd may differ according to the number of birds. These swarms may come together in different seasons and may be formed by the coming together of different species. In a herd, the larger the flock, the more foraging opportunities and the higher the chance of spotting a predator in a timely manner. Therefore, the survival of the members of the herd is important. Each bird in the herd is called a particle. The PSO has three parameters. These are the velocity that drives motion through the problem space, the particle's best point, and the swarm's best point [46]. The steps of the algorithm are given below [47]. Initially, the particle swarm is randomly distributed, with positions x_0^{I} and initial velocities v_0^{i}.

$$x_0^{\mathrm{i}} = x_{min} + \mathrm{r}(x_{max} - x_{min}) \tag{11}$$

$$v_0^{\mathrm{i}} = \left[\frac{\left(x_{min} + \mathrm{r}(x_{max} - x_{min})\right)}{\Delta \mathrm{t}}\right. \tag{12}$$

r represents a randomly chosen number between 0 and 1, x_{max} and x_{min} represent the upper and lower limits of the design variables, respectively. In step 2, the objective function values $\mathrm{f}(x_k^i)$ are calculated using the position (x_k^i) of the relevant particles. In step 3, the optimum particle position (p_k^i) and the global optimum particle position (p_k^g) in the current k iteration is updated.

$$x_{k+1}^i = x_k^i + v_{k+1}^i \Delta t \tag{13}$$

where x_{k+1}^i, $(k+1)$. position of particle i in iteration, v_{k+1}^i is the relative velocity. In the next step, the velocity of each particle is updated. It is repeated until finally the stopping criterion is met.

2.4. Ant colony optimization

ACO has emerged as a metaheuristic method in combinatorial optimization problems. It is the algorithm developed by Dorigo et al [11, 48]. This method is inspired by ant behavior. The main purpose of the ants is to reach the food source most shortly. While foraging, they randomly search for places close to their nests. When one of the ants finds a food source, it evaluates this source in terms of quality and quantity and carries some of it to its nest. On the way back to the ant nest, it leaves a substance called chemical pheromone traces on the road route. The most important element of this technique is the pheromone chemical because it is used for communication and emphasizes the quality of the solution. This chemical is updated by the ants and represents a piece of information. The presence of a pheromone trace on a road indicates the quality of the

road and increases the probability of preference. This characteristic feature in real ants has been used in artificial ant colonies to solve integrated optimization problems.

First, the number of ants is chosen, each of which represents a possible solution to the optimization problem. Each ant is randomly assigned to each decision variable. The initial amount of pheromone (τ_0) is calculated from the equation $\tau_0 = 1/Z_{min}$. Here Z_{min} is the minimum is the value. Then each ant in the colony chooses its first design variable. And then the ants choose the decision variable values. This selection is made by a decision process based on the probability calculation given below.

$$P_{ij}(t) = \frac{\left(\tau_{ij}(t)\right)^{\alpha} * \left(v_{ij}\right)^{\beta}}{\sum_{j}^{ndiv} \left(\tau_{ij}(t)\right)^{\alpha} * \left(v_{ij}\right)^{\beta}} \tag{14}$$

where $\tau_{i,j}$is the accumulated pheromone between the 'i' and 'j' paths (i,j) is the corresponding heuristic, which is the inverse of the length of the path. $\beta$is a parameter that determines the relative importance of the pheromone with respect to distance $(\beta > 0)$

$$T(i,j) \leftarrow (1-\rho) \cdot T(i,j) + \sum(1-\rho) \cdot T(i,j) \tag{15}$$

where L is the length of the tour, ρ is a constant number and represents the evaporation rate. When $\rho = 0$, there is no evaporation.

Mathematical model of the pheromone level:

$$\Delta T_K(i,j) = \begin{Bmatrix} \frac{1}{L_K} \rightarrow \text{ if ... route ... found} \\ 0 \rightarrow \text{ otherwise} \end{Bmatrix} \tag{16}$$

$\Delta T_K(i,j)$ represents the amount of pheromone released on the road by each ant. As the ant moves from one point to another, i and j in the model represent the distance between the points. $\frac{1}{L_K}$ represents the distance between two points. calculations are made and the steps are repeated until the stopping criterion (usually the maximum number of iterations) is met [49].

2.5. Bat algorithm

BA was developed by Xin-She Yang [13], inspired by the echolocation behavior of bats. Echolocation means positioning by sound. The sound intensity and signal propagation rates produced by bats by echolocation need to be updated as the iteration progresses and approaches the desired target. In the search process, artificial bats are used, which are like real bats with their natural pulse loudness and emission rate. These artificial bats search by imitating real bats [18].

The mathematical model of the bat algorithm is as follows:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{17}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*)f_i \tag{18}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{19}$$

where $\beta \in [0,1]$ is a random vector. The following equations can be used to change the sound intensity and pulse emission rates:

$$A_i^{t+1} = \alpha A_i^t \tag{20}$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{21}$$

where α is a random number between 0 and 1 and $\gamma > 0$ is a constant First, the algorithm parameters are initialized, generated, and evaluated by the initial population, and then the best solution x_{best} in the population is determined. Then a new solution is generated. Here, virtual bats are moved relative to the search space to update the rules of the bat algorithm. The best solution is developed using random walks. The new solution is evaluated. The best solution is conditionally saved. In the last step: the best solution is found and the best available solution is updated [50].

2.6. Glow worms algorithm

GSO algorithm was developed by Krishnanand and Ghose [23]. A broad class of benchmark functions is applied to test against GSO's ability to catch multiple optima. Simulation results show the success of the algorithm in finding global points. This algorithm is very close to the hunting process of honeybees or birds. Therefore, the new strategies are inspired by ABC and PSO. In addition, although some parameters of the classical GSO algorithm are constant, the performance of the population-based algorithm significantly depends on the control parameters [24].

Initially, the initial parameters are taken ($\rho = 0.4,\ \gamma = 0.6,\ \beta = 0.08,\ n_t = 5,\ s = 0.03,\ I_0 = 5$). The m glow worm is randomly distributed in the search space of the optimization problem. The lucifer ratio of each glow-worm i is then updated.

$$\mathrm{l_i}(t) = (1-\rho)\mathrm{l_i}(t-1) + \gamma J\big(x_\mathrm{i}(t)\big) \tag{22}$$

In its environment, the number of glow worms with a higher lucifer value than its lucifer value is selected. Calculate the probability of $P_{ij}(t)$ for each glow-worm in this environment.

$$P_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k\in N_i(t)}\ l_k(t) - l_i(t)} \tag{23}$$

The surrounding j glow worm is selected with the probability $P_{ij}(t)$and moved towards the i glow worm.

$$x_i(t+1) = x_i(t) + st * \left\{\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right\} \tag{24}$$

The neighborhood range is updated. Finally, the steps are repeated until the stopping criterion is met [4].

2.7. Hunting search algorithm

The HuS algorithm inspired by predators that obtain their food by hunting other animals is developed by Mirjalili [2]. According to the HuS algorithm, the strong animal eats before the weak animal. That is, the strongest animal eats the most food, while the weakest animal starves to death. After a while, when the strong animal weakens, it is removed from the colony and a new animal is produced in its place. Thus, diversity is ensured, and the adaptation of the colony increases. This is the feature of the HuS algorithm that prevents it from getting stuck in the local optimum [28].

First, the method parameters are determined. HGS is the hunting group size. MML is the maximum movement toward the leader, $HGCR$ is the hunting group consideration rate (between 0 and 1). Depending on the number of hunters (HGS), the first position of the hunters in the group is determined. Hunters' positions are updated as they move toward the leader. The new positions of hunters (new solution vectors) $x' = (x'_1, x'_2, \dots, x'_N)$.

$$x'_i = x_i + \mathrm{rand} * MML * \left(x_i^l - x_i\right) \tag{25}$$

Here x_i^l is the position value of the leader for the ith variable. Hunters reorganize and arrange their positions relative to each other to hunt more effectively.

$$x_i^j \leftarrow \begin{cases} x_i^j \in \{x_i^1, x_i^2, \dots, x_i^{HGS}\} \, HGCR \\ x_i^j = x_i^j \pm \text{rand}\,(1 - HGCR) \end{cases} \tag{26}$$

Hunters update their positions to avoid being stuck with the local optimum and to find the global optimum.

$$x_i' = x_i^l \pm \text{rand} * \left(x_i^{max} - x_i^{min}\right) * \alpha * \exp(-\beta * EN) \tag{27}$$

x_i' is the leader's position. As the algorithm continues, the solution converges to the optimum point. Finally, it is repeated as many as the number of iterations and the program is stopped [27]. First, benchmark function is examined in equation 5. This problem has nine design variables and thirteen constraints.

3. Design examples

In this study, a benchmark and five engineering design problems are discussed. First of all, the constraint problem that belongs to the benchmark problems is optimized. Then, the problems of I beam design, three truss design, tension and compression spring design, welded beam design, and speed reducer design are minimized. The results are presented in tabular form and supported by comparative convergence plots.

3.1. Constrained problem

$$f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{28}$$

Subject to

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0 \tag{29}$$

$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{11} - 10 \le 0 \tag{30}$$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0 \tag{31}$$

$$g_4(x) = -8x_1 + x_{10} \le 0 \tag{32}$$

$$g_5(x) = -8x_2 + x_{11} \le 0 \tag{33}$$

$$g_6(x) = -8x_3 + x_{12} \le 0 \tag{34}$$

$$g_7(x) = -2x_4 - x_5 + x_{10} \le 0 \tag{35}$$

$$g_8(x) = -2x_6 - x_7 + x_{11} \le 0 \tag{36}$$

$$g_9(x) = -2x_8 - x_9 + x_{12} \le 0 \tag{37}$$

and side constraints

$$0 \le x_i \le 1 \quad (i = 1,2, \dots ,9) \tag{38}$$

$$0 \le x_j \le 100 \quad (j = 10,11,12) \tag{39}$$

$$0 \le x_{13} \le 1 \tag{40}$$

This problem has been solved with six different algorithms from metaheuristic algorithm methods, FA, RFA, PSO, ACO, BA, GSO, and HuS, the results are given in Table 1.

The design results of each algorithm are shown in Fig. 1. 20,000 iterations are made for each optimization method. The HuS algorithm reached the best solution with **-14.8901**. BA algorithm gives the worst result. When the graph is examined, it is seen that the PSO algorithm converges faster than the other algorithms. When the results are examined, the refined firefly algorithm gives better results than the basic firefly algorithm.

Table 1. Optimum designs of benchmark problems obtained by six metaheuristic techniques

	FA	RFA	PSO	ACO	BA	GSO	HuS
x_1	0.99636	0.99002	0.26895	0.22563	0.78663	1.0000	0.99552
x_2	0.99969	0.99641	1.00000	0.26574	0.64589	0.99999	0.99973
x_3	0.99543	0.99509	0.99820	0.26214	0.61966	0.99110	0.99864
x_4	0.99953	0.99788	1.00000	0.68848	0.81990	0.99970	1.00000
x_5	0.99474	0.99889	1.00000	0.86697	0.44664	0.18530	0.99597
x_6	0.97393	0.99755	0.99644	0.71616	0.79828	0.99999	0.99930
x_7	0.99653	0.99706	0.99996	0.35983	0.76006	0.99999	0.99759
x_8	0.99391	0.99787	0.99608	0.19194	0.96256	1.00000	0.99597
x_9	0.99131	0.99578	1.00000	0.00826	0.92830	0.99999	0.97746
x_{10}	2.94661	2.96271	2.15160	0.69420	0.66918	0.0986	2.99598
x_{11}	2.92662	2.96974	2.99280	1.24177	2.08203	2.99934	2.99623
x_{12}	2.97210	2.96176	2.99210	0.08761	1.778533	2.99870	2.96939
x_{13}	0.99559	0.99118	0.99990	0.50179	0.51345	0.79111	0.99259
$g_1(x)$	-0.13464	-0.09469	-2.317583	-4.59703	-4.38374	-2.90204	-0.01727
$g_2(x)$	-0.14316	-0.10529	-2.321038	-4.35250	-4.43618	-2.91979	-0.01945
$g_3(x)$	-0.11102	-0.08547	-0.0184392	-3.450495	-3.60833	-0.01966	-0.03760
$g_4(x)$	-5.02433	-4.95745	0.0000009	-6.90426	-5.62389	-7.90138	-4.96818
$g_5(x)$	-5.07093	-5.00156	-5.007153	-5.76965	-3.08506	-5.00065	-5.00167
$g_6(x)$	-4.99134	-4.99903	-4.99401	-6.06107	-3.17878	-4.93022	-5.01978
$g_7(x)$	-0.04719	-0.031957	-0.848344	-2.02906	-1.41726	-4.93022	0.0000051
$g_8(x)$	-0.01776	-0.022422	-0.00000024	-0.22380	-0.27458	-0.00057	0.00000196
$g_9(x)$	-0.00703	-0.029766	0.000000834	-0.25247	-1.07490	-0.00124	0.00000578
$f(x)$	**-14.74665**	**-14.77034**	**-13.13744**	**-6.26909**	**-5.03953**	**-11.02778**	**-14.8901**

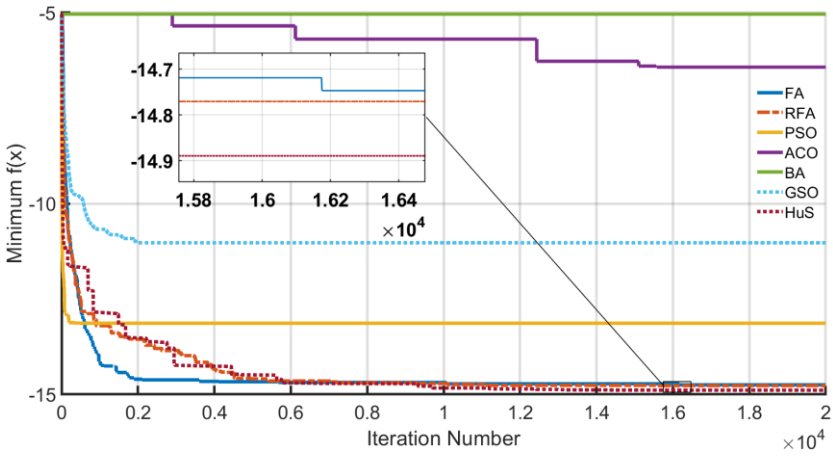

Fig.1. Convergence graph for constrained problem

3.2. Engineering design problem

FA, RFA, PSO, ACO, BA, GSO, and HuS algorithms are evaluated on some of the engineering problems, I beam design, three truss design, the tension/compression spring design problem, and the welded beam design, speed reducer design. In these problems, the number of iterations is determined as 20000. The aim of all problems is to calculate the minimization. The variables that should be used in the design are determined through algorithms.

3.2.1. I beam design

The I beam design problem which is given in Fig. 2 has a four-variable design problem. This problem is derived from a problem reported in Ref. [51]. It should be provided under the given loads under cross-sectional area and stress constraints. The aim of this study is to minimize the vertical deflection of $f(x) = PL^3/48EI$ when the beam length is 5200 cm and the modulus of elasticity is 523.104 kN/cm^2

The objective function of the problem is

$$f(b,h,t_w,t_f) = \frac{5000}{\frac{t_w(h-2t_f)}{12} + \frac{bt_f^3}{6} + 2bt_f\left(\frac{h-t_f}{2}\right)^2} \tag{41}$$

Subject to

$$g_1 = 2bt_w + t_w(h-2t_f) \leqslant 300 \tag{42}$$

$$g_2 = \frac{18h \times 10^4}{t_w(h-2t_f)^3 + 2bt_w\left(4t_f^2 + 3h(h-2t_f)\right)} + \frac{15b \times 10^3}{(h-2t_f)t_w^3 + 2t_w b^3} \leqslant 56 \tag{43}$$

where the initial design range

$$10 \leqslant b \leqslant 50,\ 10 \leqslant h \leqslant 80,\ 0.9 \leqslant t_w \leqslant 5,\ \text{and } 0.9 \leqslant t_f \leqslant 5 \tag{44}$$

There are four variables in the problem: section width (b), section height (h), body thickness (t_w) and head width (t_f). In the problem, a solution is made for 20000 iterations. The problem is solved by six different optimization algorithms and the refined firefly algorithm, and the comparison results are given in Table 2. Accordingly, it is seen that the ACO algorithm performs better than other algorithms by finding the smallest value among the other results. In the I beam design problem; it is seen that the ACO algorithm gives the best solution of 0.00003227. The BA algorithm has the worst solution with 0.00720829. Examining the convergence graphs in Figure 3 shows that ACO performs better for this problem. The performance of the RFA algorithm is better than the basic FA algorithm.

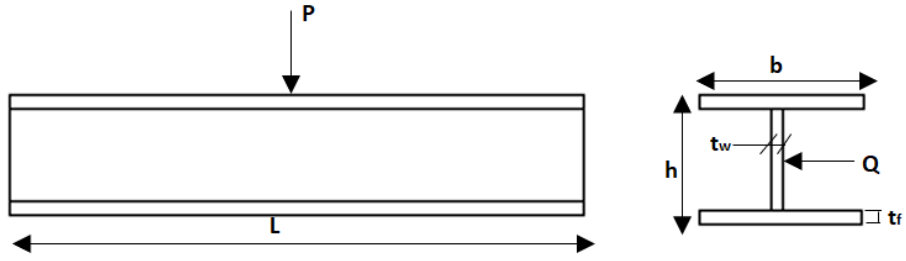

Fig.2. I-beam design

Table 2. Optimum design of I-beam design problem

	FA	RFA	PSO	ACO	BA	GSO	HuS
b (cm)	49.99978	49.9995	50.000	42.3394	49.6395	50.00	50.00
h (cm)	79.99840	79.9990	80.000	169.1580	79.8075	79.999	80.00
$t_w(cm)$	3.34765	3.89054	4.4837	42.0219	1.82301	5.0	4.99894
t_f(cm)	4.99996	4.99994	5.000	11.77813	4.984499	5.0	5.0
$g_1(x)$	-52.9907	-505.778	-582.89	-21.4476	-238.600	-650.0001	-649.863
$g_2(x)$	-435.201	-53.4115	-53.754	-55.8421	-50.41260	-53.9875	-53.9871
f(x)	**0.0071007**	**0.0071006**	**0.007100**	**0.000032**	**0.00720829**	**0.0071003**	**0.0071003**

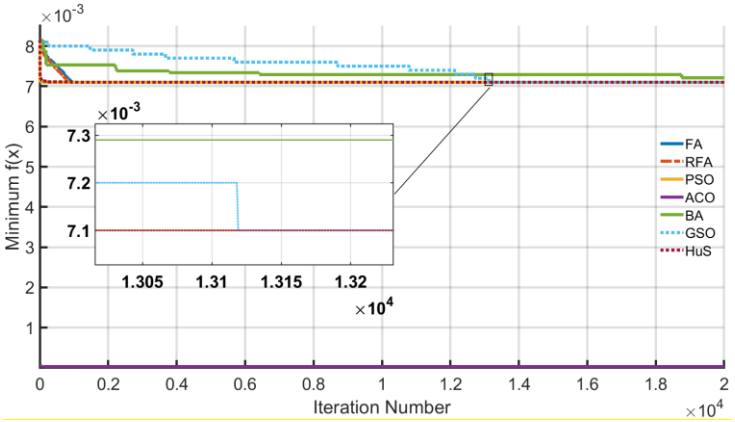

Fig.3. I-beam design convergence graph

3.2.2. Three truss design problem

The three-truss design problem which is shown in Fig. 4 is one of the civil engineering test problems for constrained algorithms. The problem has three constraints and two variables. Two parameters should be manipulated to minimize the weight subjected to stress, bending, and buckling constraints [52]. The problem definition and constraints are given in Fig. 4.

Minimize function

$$f(x) = \left(2\sqrt{2}x_1 + x_2\right) \times l \tag{45}$$

Subject to

$$g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leqslant 0 \tag{46}$$

$$g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leqslant 0 \tag{47}$$

$$g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \leqslant 0 \tag{48}$$

and side constraints

$$0 \leqslant x_i \leqslant 1 \quad , \quad i = 1,2 \tag{49}$$

The constants are taken as $l = 100\ \text{cm}, P = 2\frac{\text{kN}}{\text{cm}^2}, \sigma = 2\frac{\text{kN}}{\text{cm}^2}$

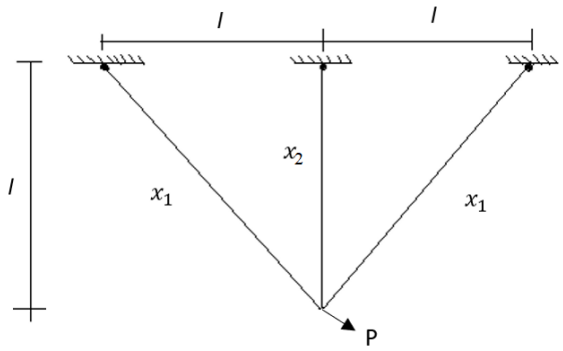

Fig.4. Three-truss design

Table 3. Optimum designs of three-truss design problem

	FA	RFA	PSO	ACO	BA	GSO	HuS
x_1	0.787954	0.788773	0.789074	0.913244	0.787210	0.78763	0.788497
x_2	0.410290	0.40797	0.407119	0.979743	0.412859	0.411240	0.408738
$g_1(x)$	0.0000008	-0.000001	0.0000009	-0.000048	-0.000339	-0.000021	0.00000999
$g_2(x)$	-1.461783	-1.46441	-1.465385	-1.455843	-1.459043	-1.460716	-1.463540
$g_3(x)$	-0.538216	-0.53559	-0.534614	-0.544205	-0.541297	-0.539305	-0.53645
f(x)	**263.8961**	**263.8960**	**263.8958**	**263.9071**	**263.9425**	**263.8994**	**263.8945**

In the three truss design problem, the HuS algorithm gives the best result with a minimum of 263.8945, while the BA algorithm gives the worst result with 263.9425. As seen in Fig. 5, the HuS Algorithm performs better. The performance of the proposed algorithm is better than the basic FA algorithm.

3.2.3. Tension/compression spring design problem

The purpose of the tension-compression spring problem in Fig. 6 is to minimize its weight. The problem has three design variables: wire diameter (d), average coil diameter (D), and the number of active coils (L).

The tension/ compression spring design function is examined in Eq. 50. The engineering design problem has three design variables and four constraints [53]. The problem definition and constraints are as follows;

$$f(x) = (x_3 + 2)x_2{x_1}^2 \tag{50}$$

Subject to

$$g_1(x) = 1 - \frac{{x_2}^3 x_3}{71785{x_1}^4} \leq 0 \tag{51}$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566({x_2}{x_1}^3 - {x_1}^4)} + \frac{1}{5108{x_1}^2} - 1 \leq 0 \tag{52}$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \tag{53}$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{54}$$

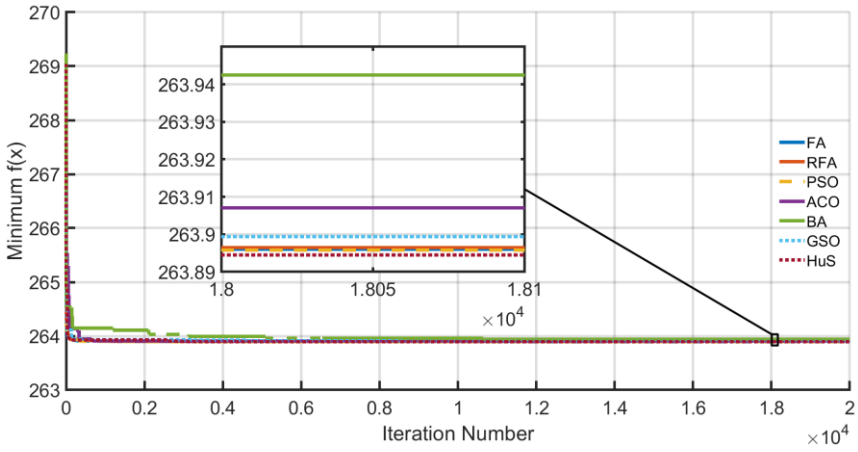

Fig.5. Three-Truss design convergence graph

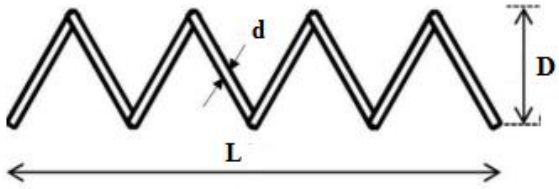

Fig.6. Tension/compression spring design [52]

and side constraints

$$0.05 \leq x_1 \leq 2 \tag{55}$$

$$0.25 \leq x_2 \leq 1.3 \tag{56}$$

$$2 \leq x_3 \leq 15 \tag{57}$$

The results of the algorithms for solving the problem are given in Table 4. The problem is shown as wire diameter d, average coil diameter D, and variable active coil number N as the variables x_1, x_2, x_3, respectively. These three design variables are calculated according to the algorithms and the best result is obtained.

It is seen that while the PSO algorithm gives the best solution value with 0.012883, the GSO algorithm gives the worst results with 0.0200664. As seen from the results in Table 4 and the convergence graph in Fig. 7, the RFA algorithm also gives better results than the FA algorithm in this problem. The PSO algorithm gave the best results for the variables. Optimum values of the three design variables: wire diameter (d) is 0.05523, average coil diameter (D) is 0.44814, and active coil number (N) is 7.42340.

3.2.4. Welded Beam Design Problem

A welded beam design is investigated in Fig. 8. The design aims to minimize the production cost. There are four different variables in the problem. These variables are expressed as weld thickness (h), weld length (l), beam width (t) and beam thickness (b) [54].

Table 4. Optimum designs of tension/compression spring design problem

	FA	RFA	PSO	ACO	BA	GSO	HuS
x_1	0.05568	0.055895	0.05523	0.09004	0.05489	0.06674	0.05660
x_2	0.46062	0.466573	0.44814	1.27178	0.43444	0.75330	0.48679
x_3	7.10099	6.907491	7.42340	5.92907	8.27834	3.97967	6.38659
$g_1(x)$	-0.005504	-.00124933	0.000000959	-0.00678	-0.04124	-0.19422	0.000009943
$g_2(x)$	-0.000091	-.00006284	0.000000837	-0.02391	-0.00839	-0.09075	0.000009993
$g_3(x)$	-4.19095	-4.2208090	-4.20344	-3.92445	-3.93469	-3.15091	-4.25266
$g_4(x)$	-0.65579	-.65168760	0.66441	-0.71506	-0.67378	-0.45331	-0.63774
$f(x)$	**0.0129988**	**0.0129845**	**0.012883**	**0.013349**	**0.013456**	**0.0200664**	**0.013079**

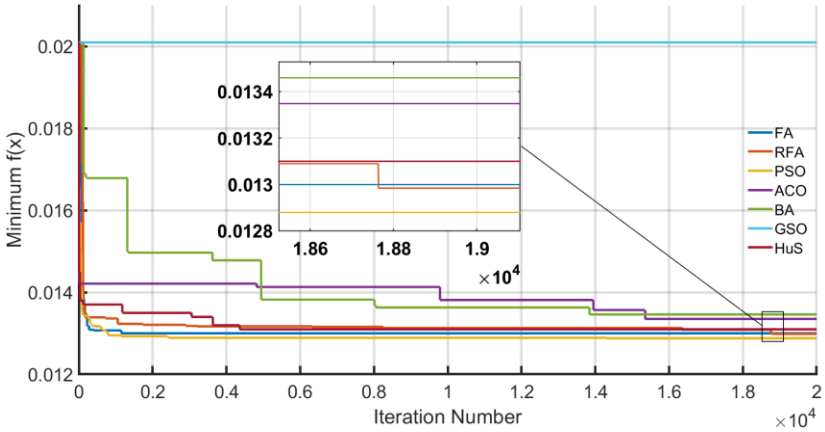

Fig.7. Design history graph for tension/compression spring design

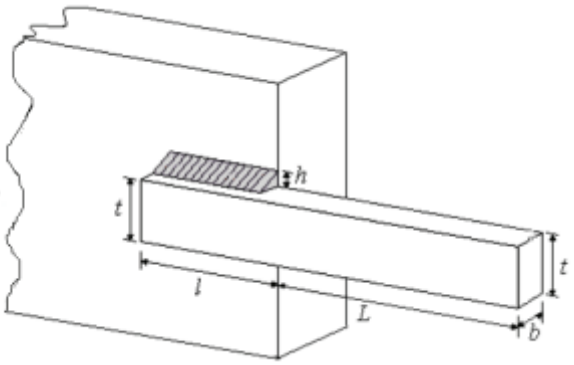

Fig.8. Welded Beam Design [2]

The welded beam design problem function is examined in Eq. 58. The engineering design problem has four design variables and seven constraints. The problem definition and constraints are as follows:

Minimize

$$f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{58}$$

Subject to

$$g_1(x) = \tau(x) - \tau_{max} \leq 0 \tag{59}$$

$$g_2(x) = \sigma(x) - \sigma_{max} \leq 0 \tag{60}$$

$$g_3(x) = \delta(x) - \delta_{max} \leq 0 \tag{61}$$

$$g_4(x) = x_1 - x_4 \leq 0 \tag{62}$$

$$g_5(x) = P - P_c(x) \leq 0 \tag{63}$$

$$g_6(x) = 0.125 - x_1 \leq 0 \tag{64}$$

$$g_7(x) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \tag{65}$$

Variable ranges

$$0.1 \leq x_1 \leq 2 \tag{66}$$

$$0.1 \leq x_2 \leq 10 \tag{67}$$

$$0.1 \leq x_3 \leq 10 \tag{68}$$

$$0.1 \leq x_4 \leq 2 \tag{69}$$

where

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{70}$$

$$\tau' = \frac{P}{\sqrt{2x_1x_2}}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right) \tag{71}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{72}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{73}$$

$$\sigma(x) = \frac{6PL}{x_4x_3^2}, \delta(x) = \frac{4PL^3}{Ex_3^3x_4} \tag{74}$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{75}$$

The welded beam is made of steel and supports 6000 pound-force. The modulus of elasticity (E) is 30×10^6psi, shear modulus (G) is 12×10^6psi, beam length (L) is 14 in shear stress value is (τ_{max}) 30,000 psi and normal stress value (σ_{max}) is 30,000 psi [55]. This welded beam problem is solved by using six different algorithm methods. The results of the algorithms for solving the problem are given in Table 5.

Table 5. Optimum designs of welded beam problem

	FA	RFA	PSO	ACO	BA	GSO	HuS
x_1	0.19905	0.201390	0.24348	0.72599	0.21986	0.21445	0.2067
x_2	3.64112	3.602030	3.04932	3.80475	3.50224	3.37779	3.45843
x_3	9.04015	9.041869	8.30663	5.29021	8.58467	8.86544	9.0156
x_4	0.20595	0.205819	0.24348	0.867295	0.23372	0.215460	0.2067
$g_1(x)$	-62.9463	-103.74800	-0.00098	-519.599	357.1807	-33.9287	-0.00391
$g_2(x)$	-55.04883	-47.818360	0.00000	-1075.896	738.662	-237.914	-0.00195
$g_3(x)$	-0.00689	-0.0044293	0.00000098	-0.03838	0.01385	-0.00101	0.0000099
$g_4(x)$	-3.4157	-3.419805	-3.33487	-3.2716	3.305503	-3.39821	-3.43038
$g_5(x)$	-0.074055	-0.0763898	-0.118479	-0.050592	0.9486	-0.08945	-0.081701
$g_6(x)$	-0.23557	-0.2355717	-0.23427	-0.23604	-0.23515	-0.23537	-0.23551
$g_7(x)$	-20.5762	-10.12109	-3395.310	-743.491	2500.054	-805.556	-75.12402
$f(x)$	**1.73951**	**1.7373**	**1.85862**	**1.88087**	**1.87647**	**1.76858**	**1.72838**

There are four variables in the problem: weld thickness (h), weld length (l), beam width (t), and beam thickness (b). The problem is solved with six different optimization algorithms and refined firefly algorithm. The comparison results and the convergence graph are given in Table 5 and Fig. 9, respectively. Accordingly, it is seen that the HuS algorithm performs better than other algorithms by finding the smallest value in terms of best, worst, and average results. The algorithms that follow HuS in terms of finding the best value are FA and GSO, respectively. In order not to get stuck in the local optimum, the proposed RFA algorithm gives better results than the FA algorithm. ACO showed the worst performance for this problem. The optimum values found by the algorithms for the variables of the welded beam problem are presented in Table 5. Accordingly, the optimum values found for $h, l, t,$ and b values by HuS, which found the lowest cost, are 0.206, 3.45, 9.01, and 0.2061, respectively.

3.2.5. Speed reducer design problem

The weight of the speed reducer which is shown in Fig. 10 will be minimized by using stresses, deflections, and constraints. The variables are x_1 face width, x_2 the module of teeth, x_3 number of teeth in the pinion, x_4 length of the first shaft between bearings, x_5 length of the second shaft between bearings, and x_6 the diameter of the first and x_7 second shafts.

The problem has 11 constraints [53]. The problem definition and constraints are as follows:

$$f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2) \tag{76}$$

Subjected to

$$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leqslant 0 \tag{77}$$

$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leqslant 0 \tag{78}$$

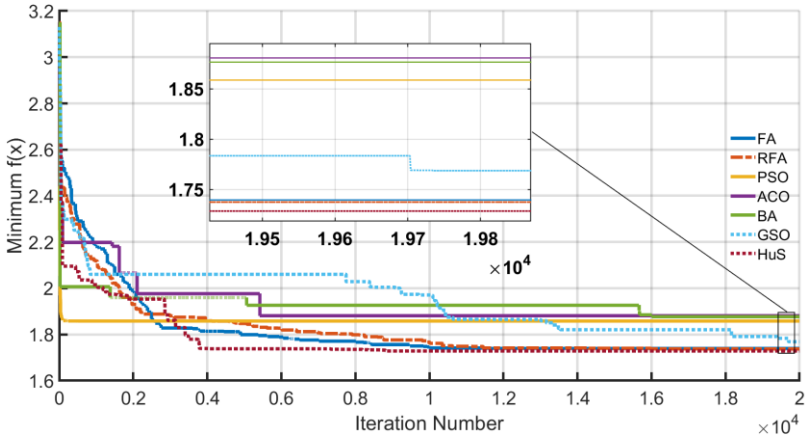

Fig. 9. Welded-beam design convergence graph

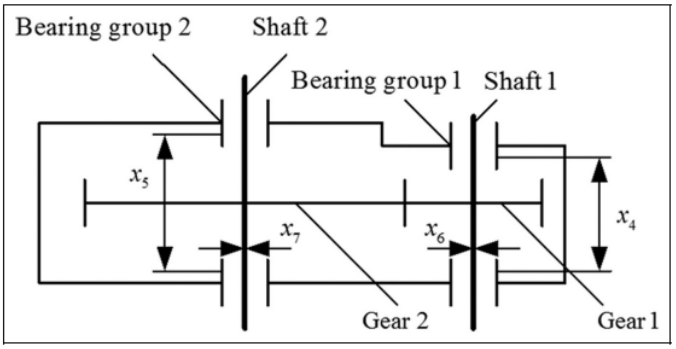

Fig. 10. Design variables of the speed reducer design [56]

$$g_3(x) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \leqslant 0 \tag{79}$$

$$g_4(x) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \leqslant 0 \tag{80}$$

$$g_5(x) = \frac{\left[\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right]^{\frac{1}{2}}}{110x_6^3} - 1 \leqslant 0 \tag{81}$$

$$g_6(x) = \frac{\left[\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6\right]^{\frac{1}{2}}}{85x_7^3} - 1 \leqslant 0 \tag{82}$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \leqslant 0 \tag{83}$$

$$g_8(x) = \frac{5x_2}{x_1} - 1 \leqslant 0 \tag{84}$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \leqslant 0 \tag{85}$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leqslant 0 \tag{86}$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leqslant 0 \tag{87}$$

where

$$2.6 \leqslant x_1 \leqslant 3.6,\ 0.7 \leqslant x_2 \leqslant 0.8,\ 17 \leqslant x_3 \leqslant 28,\ 7.3 \leqslant x_4 \leqslant 8.3,$$
$$7.3 \leqslant x_5 \leqslant 8.3,\ 2.9 \leqslant x_6 \leqslant 3.9,\ 5.0 \leqslant x_7 \leqslant 5.5$$

Table 6. Optimum designs of speed reduce design problem

	FA	RFA	PSO	ACO	BA	GSO	HuS
x_1	3.50581	3.501651	3.5000	3.561687	3.508327	3.500003	3.499966
x_2	0.700201	0.700018	0.7000	0.709687	0.701039	0.700000	0.70000
x_3	17.00031	17.0014	17.000	20.35658	17.044630	17.00206	17.00006
x_4	7.31221	7.58337	7.3003	7.470057	8.029846	8.298532	7.300121
x_5	7.765204	7.772187	7.7155	8.031518	7.861442	7.717353	7.715663
x_6	3.355170	3.354811	3.3502	3.501754	3.358119	3.352268	3.350214
x_7	5.288564	5.287802	5.2866	5.340103	5.314514	5.286745	5.286650
$g_1(x)$	-0.0759989	-0.074477	-0.07393	-0.079102	-0.0812605	-0.074028	-0.07390939
$g_2(x)$	-0.1998175	-0.198553	-0.19801	-0.203407	-0.2064430	-0.198193	-0.19799620
$g_3(x)$	-0.4997753	-0.441685	-0.49917	0.529438	-0.3423945	-0.266151	-0.49914860
$g_4(x)$	-0.9029523	-0.902615	-0.9046382	-0.896360	-0.9016261	-0.904586	-0.9046311
$g_5(x)$	-0.0044078	-0.003624	0.0000007	-0.045000	-0.0058325	-0.000059	0.00000097
$g_6(x)$	-0.0010735	-0.000640	-0.0000089	-0.007493	-0.0156226	-0.000050	0.00000276
$g_7(x)$	-0.702409	-0.702467	-0.7024981	-0.701786	-0.7012764	-0.702464	-0.70249900
$g_8(x)$	-0.0013701	-0.000446	0.0000009	-0.000732	-0.0008933	-0.000000	0.00000979
$g_9(x)$	-0.7954355	-0.795731	-0.7958309	-0.795174	-0.7950440	0.795833	-0.7958353
$g_{10}(x)$	-0.0518932	-0.085867	-0.0513276	-0.041138	-0.1360759	-0.165105	-0.05134165
$g_{11}(x)$	-0.0061536	-0.007154	-0.0000238	-0.029581	-0.0146889	-0.000250	-0.00004519
$f(x)$	**3001.400**	**3001.105**	**2994.513**	**3031.992**	**3039.720**	**3004.267**	**2994.4730**

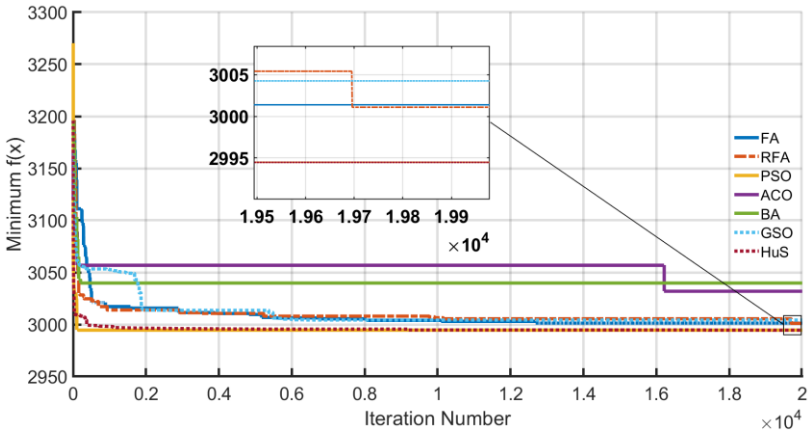

Fig. 11. Speed Reduce design convergence graph

As seen in Table 6, the HuS algorithm gives the best result for speed reduce design problems with 2994.4730. The algorithms that follow HuS in terms of finding the best value are PSO, RFA, and FA, respectively. BA algorithm gives the worst result with 3039.720. When Fig. 11 is examined, it is seen that the performance of the PSO algorithm is better. RFA algorithm gives better results than the FA algorithm. However, it is worse than the HuS algorithm.

4. Conclusion

The solution capabilities of meta-heuristic algorithms, which are frequently used in recent years, in engineering applications may differ. Although metaheuristic algorithms are generally successful in optimization problems, the problem structure in which each algorithm comes to the fore may vary in terms of performance. In this study, the minimization performances of FA, RFA, PSO, ACO, BA, GSO, and HuS algorithms are compared for a mathematical and five different engineering problems. When the I beam design problem is examined, the ACO algorithm gives the best solution, while the algorithm that gives the best results in other problems is not ACO. However, in the tension and compression spring design problem, the best result is obtained from the PSO algorithm. For the other three engineering problems, the algorithm that gives the best results is HuS. On the other hand, the ACO algorithm gives the worst result in the welded beam design problem. In the tension and compression spring design problem, the worst result is obtained from the GSO algorithm. In the other three problems, the worst results are for the BA algorithm. On the other hand, the firefly algorithm is improved to get better results in solving engineering design problems in this study. It is seen that the refinement design algorithm is mathematically quite simple to adapt to the problems, but it is more effective than the simple FA algorithm in finding the solutions to the problems. Position update is done so that the RFA algorithm can solve without being stuck to the local. As a result of the improvement, it is seen that the RFA algorithm gives better results than the FA algorithm in all problems. In more extensive problems, the effectiveness of the improvement made by solving the RFA algorithm can be investigated by further studies.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

[1] Çelik Y, Yıldız İ, Karadeniz AT (2019) A brief review of metaheuristic algorithms improved in the last three years. European Journal of Science and Technology 463–477 (in Turkish).

[2] Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Advances in Engineering Software 69:46–61

[3] Lam HT, Nicolaevna PN, Quan NTM (2007) A heuristic particle swarm optimization. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 174–174.

[4] Wu B, Qian C, Ni W, Fan S (2012) The improvement of glow worms swarm optimization for continuous optimization problems. Expert Systems with Applications 39:6335–6342.

[5] Hamdan S, El Zawawi A (2009) Testing of a modified particle swarm optimization algorithm using different benchmark functions. In: Proceedings of the 3rd International Workshop on Soft Computing Applications, pp. 115–118.

[6] Uriarte A, Melin P, Valdez F (2016) An improved particle swarm optimization algorithm applied to benchmark functions. In Proceedings of the 8th International Conference on Intelligent Systems, pp. 128–132.

[7] Bhushan B, Pillai SS (2013) Particle swarm optimization and firefly algorithm: performance analysis. In: Proceedings of the 3rd International Advance Computing Conference, pp. 746–751.

[8] Tiwari A, Ahuja K, Yadav A, Bansal JC, Deep K, Nagar AK (2021) Soft Computing for Problem Solving: Proceedings of SocProS 2020, Volume 1. Springer, Singapore, Singapore.

[9] Blum C (2005) Ant colony optimization: Introduction and recent trends. Physics of Life Reviews 2:353–373.

[10] Maniezzo V, Gambardella LM, de Luigi F (2004) Ant Colony Optimization. In: Onwubolu GC, Babu BV (eds) New Optimization Techniques in Engineering. Springer, Berlin, Heidelberg, pp. 101–121.

[11] Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In: Proceedings of the Congress on Evolutionary Computation (Cat. No. 99TH8406), pp. 1470–1477.

[12] Dorigo M, Stützle T (2019) Ant Colony Optimization: Overview and Recent Advances. In: Gendreau M, Potvin J-Y (eds) Handbook of Metaheuristics. Springer International Publishing, Cham, pp. 311–351.

[13] Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer, pp. 65–74.

[14] Mirjalili S, Mirjalili SM, Yang X-S (2014) Binary bat algorithm. Neural Comput & Appl 25:663–681.

[15] Cui Z, Li F, Zhang W (2019) Bat algorithm with principal component analysis. Int J Mach Learn & Cyber 10:603–622.

[16] Gandomi AH, Yang X-S, Alavi AH, Talatahari S (2013) Bat algorithm for constrained optimization tasks. Neural Comput & Appl 22:1239–1255.

[17] Tsai PW, Pan JS, Liao BY, Tsai MJ, Istanda V (2012) Bat algorithm inspired algorithm for solving numerical optimization problems. Applied Mechanics and Materials 148–149:134–137.

[18] Yang X-S, He X (2013) Bat algorithm: Literature review and applications. International Journal of Bio-Inspired Computation 5:141–149.

[19] Lukasik S, Żak S (2009) Firefly algorithm for continuous constrained optimization tasks. In: International Conference on Computational Collective Intelligence. Springer, pp 97–106.

[20] Carbas S (2016) Design optimization of steel frames using an enhanced firefly algorithm. Engineering Optimization 48:2007–2025.

[21] Farahani ShM, Abshouri AA, Nasiri B, Meybodi MR (2011) A Gaussian firefly algorithm. International Journal of Machine Learning and Computing 1:448–453.

[22] Yang X-S, He X (2013) Firefly algorithm: Recent advances and applications. International Journal of Swarm Intelligence 1:36–50.

[23] Krishnanand KN, Ghose D (2006) Glow worms swarm-based optimization algorithm for multimodal functions with collective robotics applications. Multiagent and Grid Systems 2:209–222.

[24] Krishnanand KN, Ghose D (2009) Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intelligence 3:87–124.

[25] Dogan E, Ozyuksel Ciftcioglu A (2020) Weight optimization of steel frames with cellular beams through improved hunting search algorithm. Advances in Structural Engineering 23:1024–1037.

[26] Oftadeh R, Mahjoob MJ, Shariatpanahi M (2010) A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. Computers & Mathematics with Applications 60:2087–2098

[27] Oftadeh R, Mahjoob MJ (2009) A new meta-heuristic optimization algorithm: Hunting Search. In: 5th International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, pp. 1–5.
[28] Özyüksel Çiftçioğlu A (2021) A Study on the performance of Grey Wolf Optimizer, pp. 100–116
[29] Çelik Y (2013) The Improving Performance of Marriage in Honeybee Optimization Algorithm (MBO) on Optimization Problems. Thesis, Selçuk University, Institute of Natural and Applied Science (in Turkish).
[30] Khalilpourazari S, Khalilpourazary S, Özyüksel Çiftçioğlu A, Weber G-W (2021) Designing energy-efficient high-precision multi-pass turning processes via robust optimization and artificial intelligence. J Intell Manuf 32:1621–1647.
[31] Moles CG, Mendes P, Banga JR (2003) Parameter estimation in biochemical pathways: a comparison of global optimization methods. Genome Research 13:2467–2474.
[32] Kim JH, Geem ZW, Kim ES (2001) Parameter estimation of the nonlinear Muskingum model using harmony search. JAWRA Journal of the American Water Resources Association 37:1131–1138.
[33] He J, Bresler M, Chiang M, Rexford J (2007) Towards robust multi-layer traffic engineering: Optimization of congestion control and routing. IEEE Journal on Selected Areas in Communications 25:868–880.
[34] Wang N, Ho KH, Pavlou G, Howarth M (2008) An overview of routing optimization for internet traffic engineering. IEEE Communications Surveys & Tutorials 10:36–56.
[35] Garden RW, Engelbrecht AP (2014) Analysis and classification of optimisation benchmark functions and benchmark suites. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1641–1649.
[36] Coello CAC (2000) Treating constraints as objectives for single-objective evolutionary optimization. Engineering Optimization 32:275–308.
[37] Altunbey Özbay F, Özbay E (2021) Performance analysis of seagull optimization algorithm for constrained engineering design problems. Adiyaman University Journal of Engineering Sciences 15:469-485 (in Turkish).
[38] Jamil M, Yang X-S (2013) A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation 4:150–194.
[39] Carbas S, Artar M (2021) Optimum design of cold-formed steel frames via five novel nature-inspired metaheuristic algorithms under consideration of seismic loading. Structures 33:4011–4030.
[40] Özyüksel Çiftçioğlu A, Doğan E (2017) Comparison of the performance of current optimization techniques in the solution of mathematical problems. Celal Bayar University Journal of Sciences 13(2):579-591 (in Turkish).
[41] Şeker S, Doğan E (2012) Performance testing of hunting search algorithm in finding the optimum solution of engineering design problems. In: Proceedings of the 7th International Symposium on Civil and Environmental Engineering, pp. 1.
[42] Çiftçioğlu AÖ, Doğan E (2019) Optimum design of steel frames using different stochastic techniques. Konya Journal of Engineering Sciences 7(4):847-861 (in Turkish).
[43] Yang X-S (2010) Nature-Inspired Metaheuristic Algorithms. Luniver Press.
[44] Dogan E, Hasancebi O, Polat SM (2009) A refinement of discrete particle swarm optimization for large-scale truss structures. Asian Journal of Civil Engineering 10:321-334.
[45] Bansal JC, Pal NR (2019) Swarm and evolutionary computation. Evolutionary and Swarm Intelligence Algorithms 1–9.
[46] Eser O, Çakan A, Kalyoncu M, Botsali F (2021) Determining the quarter vehicle model design parameters using bees algorithm (BA) and particle swarm optimization (PSO). Konya Journal of Engineering Sciences 9(3):621–632 (in Turkish).
[47] Saka MP, Dogan E, Aydogdu I (2013) Analysis of swarm-intelligence based algorithms for constrained optimization. In: Swarm Intelligence and Bio-Inspired Computation pp. 25-48.
[48] Dorigo M, Di Caro G, Gambardella LM (1999) Ant algorithms for discrete optimization. Artificial Life 5:137–172.
[49] Chandra Mohan B, Baskaran R (2012) A survey: Ant Colony Optimization based recent research and implementation on several engineering domain. Expert Systems with Applications 39:4618–4627.
[50] Fister I, Yang X-S, Fong S, Zhuang Y (2014) Bat algorithm: Recent advances. In: 15th International Symposium on Computational Intelligence and Informatics, pp. 163–167.

[51] Cheng M-Y, Prayogo D (2014) Symbiotic Organisms Search: A new metaheuristic optimization algorithm. Computers & Structures 139:98–112.
[52] Chen H, Xu Y, Wang M, Zhao X (2019) A balanced whale optimization algorithm for constrained engineering design problems. Applied Mathematical Modelling 71:45–59.
[53] Arora JS (1989) Introduction to Optimum Design. McGraw-Hill, New York, USA.
[54] Coello Coello CA (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry 41:113–127.
[55] Li Y, Lin X, Liu J (2021) An improved gray wolf optimization algorithm to solve engineering problems. Sustainability 13(6):3208.
[56] Yuan R, Li H, Gong Z, et al (2017) An enhanced Monte Carlo simulation–based design and optimization method and its application in the speed reducer design. Advances in Mechanical Engineering 9:1687814017728648
[57] Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems. Computers & Structures 110:151–166.