IJST, Transactions of Mechanical Engineering, Vol. 38, No. M2, pp 403-421
Printed in The Islamic Republic of Iran, 2014

A MODIFIED FIREFLY ALGORITHM FOR ENGINEERING DESIGN OPTIMIZATION PROBLEMS*

M. J. KAZEMZADEH-PARSI

Dept. of Mechanical Engineering, Shiraz Branch, Islamic Azad University, Shiraz, I. R. of Iran
Email: kazemzadeh@iaushiraz.ac.ir

Abstract– In the present study, some modifications on the firefly algorithm are presented to improve its performance. The firefly algorithm is a recently developed robust metaheuristic optimization technique which mimics the social behavior of fireflies based on their flashing characteristics. To improve its performance three basic modifications are proposed in the present work. These modifications consist of adding memory, adding newborn fireflies and proposing a new updating formula. To evaluate the applicability of the proposed method, three classical engineering design optimization problems and three sizing optimization of truss structures are solved and results are compared with those available in the literature. It is observed that the proposed method can effectively be used in solution of engineering design optimization problems.

Keywords– Metaheuristic optimization, firefly algorithm, engineering optimization, truss structures, sizing optimization

1. INTRODUCTION

Optimization appears in many real world problems, e.g. engineering, agricultural, and manufacturing, economics, management, medicine, etc. There are also other problems which are not optimized inherently, but can be reformulated as an optimization problem; e.g. inverse problems [1]. Therefore, different techniques are being extensively used for solution of such problems in various spheres of human activities. The unified approach is to represent the problem with a mathematical model at first and then invoke an appropriate optimization method to solve the problem and obtain the unknowns in such a way that satisfies all of our desires. As there are many inherently different optimization problems, there is no single optimization method that can be used in all of the problems. Therefore many different methods with special capabilities are developed to handle different problems.

In the case of engineering design, most of the optimization problems are highly nonlinear including many design variables and complicated constraints. In addition, not all design variables are continuous and some variables can only take certain discrete values. As a result, the optimization problem often has a complex feasible domain with multiple local optima which requires problem specific techniques and there is no guarantee the global optima will be found.

Significant amount of work has been done in developing efficient methods for solving optimization problems. Many classical optimization algorithms use the gradient information and normally work well for smooth problems; however, if there is some discontinuity in the objective function, constraints or design variables they may not work. In addition, these classical methods may converge to local optimum points. Thus, to overcome these difficulties, the gradient free algorithms are preferred.

One class of derivative free techniques consists of evolutionary or nature inspired metaheuristic optimization algorithms. The vast majority of these algorithms have been derived from the behavior of

*Received by the editors September 6, 2013; Accepted March 12, 2014.

biological or physical systems in nature, for example, biological evolution, stellar evolution, thermal annealing, animal behavior, music improvisation, etc. [2]. Two important characteristics of metaheuristic optimization methods are intensification and diversification [3]. Searching the current solutions to find the local optimums and selecting the best candidate designs is known as intensification. Diversification, on the other hand, means that optimization algorithm explores the entire search space for global optima. These techniques have gained a lot of popularity in recent years because of their ability to deal with complex optimization problems which are otherwise difficult to solve.

Various bio inspired optimization algorithms have been presented in literature. The most popular methods are genetic algorithm [4], evolutionary strategies [5], evolutionary programming [6], particle swarm optimization [7], differential evolution [8], ant colony optimization [9], honey bee algorithm [10], bee algorithm [11], artificial bee colony [12], cuckoo search [13], hunting search [14], bat algorithm [15] and firefly algorithm [3]. Besides bio inspired algorithms, there are the nature inspired algorithms that mimic physical phenomena such as simulated annealing [16], harmony search [17], big bang-big crunch [18], charged system search [19], spiral optimization [20], biogeography based optimization [21], krill herd algorithm [22], teaching learning algorithm [23] and ray optimization [24].

Firefly algorithm is a recently developed, promising, metaheuristic optimization technique originally proposed by Yang [3]. The FA is based on the idealized behavior of the flashing characteristics of fireflies. Based on Yang's works, the FA is very efficient at finding the global optima with high success rates [3]. It is also shown, using various test functions, that the FA is superior to both PSO and GA in terms of both efficiency and success rate [3, 25, 26].

One of the most important things in developing or selecting a proper optimization algorithm (especially in metaheuristic optimization algorithms) is the number of function evaluations required to obtain the solution. Because, in a majority of the real world engineering design optimization problems, evaluation of objective function and constraints involve intensive numerical computations and the function evaluations may be so costly. Therefore, accurate tuning or modifying of the optimizer to reduce the number of evaluations is very crucial. To tackle this goal, in the present work, some basic modifications are introduced in the original FA.

After the first presentation of FA in [27] some modifications are proposed by different researchers; for further details refer to [28-32]. To continue this way, in the present work, three new notions are inserted in the basics of the FA to improve its performance. They are: i) adding a kind of memory to transfer some information obtained in each iteration to the next one, ii) adding newborn fireflies to extensively explore the search space for global optimum point and iii) introducing a new updating formula to reduce wandering motion of fireflies. To evaluate the applicability of proposed modifications, six numerical examples consist of three benchmark problems in engineering design optimization problems as well as three cross sectional area optimization of plane and space truss structures are solved and the results are compared with those available in the literature for different optimization algorithms. The results show that the proposed modifications can be effectively used in solution of engineering design optimization problems.

The remaining part of this article deals at first with a brief review of the basics of the FA and is followed by a detailed description of the proposed method. After a short note on the constraint handling approach, the benchmark problems are solved and the results are presented. Finally the article ends with conclusions and references.

2. FIREFLY ALGORITHM

FA is a new population based metaheuristic optimization algorithm which is inspired by the flashing behavior of fireflies. There are three idealized rules in the traditional FA [3]: (a) all fireflies are unisex so

that one firefly will be attracted to other fireflies regardless of their sex; (b) attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less brighter one will move towards the brighter one. If there is no brighter one than a particular firefly, it will move randomly; and (c) the brightness of a firefly is determined by the landscape of the objective function.

In other words, in the FA each firefly will be attracted to the brighter fireflies, and at the same time they will move randomly [3]. The attractiveness is proportional to the brightness of the flashing light which will decrease with distance. Therefore, the attractiveness will be evaluated in the eye of the other fireflies and the light absorption characteristic of the surrounding will cause reduction of light intensity and the attractiveness of the fireflies. The attractiveness β can be defined as follows.

$$\beta = \beta_0 e^{-\gamma r^2} \tag{1}$$

The light absorption coefficient γ can be considered as a constant representing a characteristic length scale of the problem. Initial light intensity β_0 is the attractiveness at $r = 0$. The Cartesian norm $r = \left\| \mathbf{x}_i - \mathbf{x}_j \right\|$ represents the distance between any two fireflies *i* and *j*. The updating formula for relocating any firefly *i* which is attracted by a brighter firefly *j* is as follows.

$$\mathbf{x}_i = \beta(\mathbf{x}_j - \mathbf{x}_i) + \alpha\boldsymbol{\varepsilon} \tag{2}$$

where the first term is due to the attraction, whereas the second term is random walk. α is the randomization parameter and $\boldsymbol{\varepsilon}$ is a random vector within the search space and can be defined as follows.

$$\boldsymbol{\varepsilon} = (\xi - 0.5)(\mathbf{U} - \mathbf{L}) \tag{3}$$

where ξ is a random number between 0 and 1; $\mathbf{U}$ and $\mathbf{L}$ are the side constraints which determine the upper and lower bounds of the design variables, respectively. Based on this, the basic steps of the FA can be summarized as a pseudo code shown in Fig. 1 [3]. It must be noted that the original FA pseudo code which has been published in [27] has some ambiguities which are corrected in the present form in Fig. 1.

Define the upper bound **U** *and lower bound* **L** *for the design variables*
Generate an initial population of fireflies $\mathbf{x}_i$ *(i=1 to n)*
Evaluate the response function F_i *for each firefly* $\mathbf{x}_i$
Sort the fireflies based on their response function
for *t=1 to Maximum iteration*
  $\mathbf{y}_i = \mathbf{x}_i$ *(i=1 to n)*
  for *i=1 to n*
    for *j=1 to n*
      if $F_i >= F_j$
        r=norm($\mathbf{x}_i$*-*$\mathbf{x}_j$*)*
        $\beta = \beta_0 \times \exp(-\gamma r^2)$
        $\boldsymbol{\varepsilon} = (rand - 0.5) \times (\mathbf{U} - \mathbf{L})$
        $\mathbf{x}_i = \mathbf{x}_i + \beta(\mathbf{y}_j - \mathbf{x}_i) + \alpha\boldsymbol{\varepsilon}$
      end if
    next *j*
    Check the side constraints for firefly $\mathbf{x}_i$
  next *i*
  Evaluate the response function F_i *for each firefly* $\mathbf{x}_i$
  Sort ascending fireflies based on their response function
  Present the first firefly as the best solution obtained in this iteration
next *t*

Fig. 1. Pseudo code of the original firefly algorithm

FA may share many similarities with PSO. In fact, it has been proved in [3] that when $\gamma \to \infty$, the FA will become an accelerated version of PSO, while in $\gamma \to 0$, the FA reduces to a version of random search algorithms.

3. MODIFIED FIREFLY ALGORITHM

In the current article, three basic modifications are introduced in the original FA to improve its performance. These points are explained in detail in the following subsections.

a) Memory

In many metaheuristic optimization algorithms there exists a kind of memory which transfers some information from one iteration to the other. For example, in the PSO a memory is defined to retain the best particle and also the global best positions. This information is then used in the next iteration for updating the particles. As another example, in the genetic algorithm the offspring inherit the genes from its parents and then transfer them to the next generation via crossover operator. Unfortunately, the traditional FA suffers from lack of memory and no specific information is transferred from one iteration to the other.

To further explain, consider a firefly which reaches to an optimum or near optimum point in one iteration. This firefly will participate in the updating process to generate the next population. It will attract other fireflies but it has no more chance to do this in successive iterations because the position of this firefly will also be changed and its information lost. To overcome this point, it is necessary to allow some information of high rank fireflies to be transferred to the next iteration. In the current work, two approaches are proposed for this purpose and are described as follows.

In the first approach, in each iteration a number of the high rank fireflies (say m_1) are directly transferred to the next iteration without any change in their position. To do this practically, in each iteration the updating operator is not applied on the first m_1 high rank fireflies and therefore the rest of them ($n - m_1$ fireflies) participate in the updating process.

In the second approach, in each iteration, a number of low rank fireflies (say m_2) are removed from the population and replaced by equal numbers of high rank fireflies from the previous iteration. To do this, in each iteration, a copy of m_2 high rank fireflies is stored and in the next iteration the updating operator applies only to the first $n - m_2$ fireflies and the m_2 low rank fireflies are replaced with the m_2 high rank fireflies stored in the previous iteration.

The first approach tends to fix the high rank fireflies and other fireflies explore the search space extensively for the global optimum point. On the other hand, in the second approach some of the low rank fireflies are removed and fewer fireflies are allowed to explore the search space for global optimum. Therefore, the second approach converges rapidly to one optimum point and begins to search intensively near the optimum points. Although the second approach converges rapidly and gives accurate results, it may be trapped in a local minimum point. It is also worth noting that there is no need to evaluate the objective function of the stored fireflies in both of these approaches. It is also clear that only one of these approaches must be chosen and using both of them simultaneously is not effective.

b) Newborn fireflies

Mutation operator is one of the appealing features of the genetic algorithm. It prevents the genetic algorithm from being trapped in a local optimum and plays the role of recovering the lost genetic materials. If crossover operator in the genetic algorithm is supposed to exploit the current solution to find better ones (intensification), mutation is supposed to help exploration of the whole search space (diversification). Therefore, the mutation operator maintains genetic diversity in the population and helps escape from local minimum trap.

Unfortunately, no such mechanism was designed in the original FA. As the second modification, a similar notion is introduced in the FA via adding newborn fireflies into the population. To tackle this, in each iteration, some new fireflies (say k) are generated randomly within the search space and inserted into the population. To maintain the total number of fireflies constant, it is necessary to remove k fireflies from the population, and this is done by removing the low ranked ones.

c) Updating formula

The updating formula of the original FA presented in Eq. (2) changes the position of each firefly towards all of the brighter fireflies in a stepwise manner regardless of the objective function of this firefly in these steps. To explain this, refer to Fig. 2 which schematically represents the updating path of a firefly in a two dimensional search space with 11 fireflies. In this figure, as in the original FA, the fireflies are sorted according to their objective functions. For example, the fireflies 1 to 5 are brighter that the firefly 6. As it is shown, using Eq. (2), the firefly 6 changes its position repetitively toward the fireflies 1 to 5 and eventually reaches its final position. It is worth noting that the objective function is not reevaluated in each step where the position of this firefly changes. Therefore, relocation of this firefly is based on its objective at its initial position. As it is shown schematically in Fig. 2, it seems that this firefly is wandering and follows a zigzag updating path. This behavior of the original FA decreases overall performance of the algorithm.

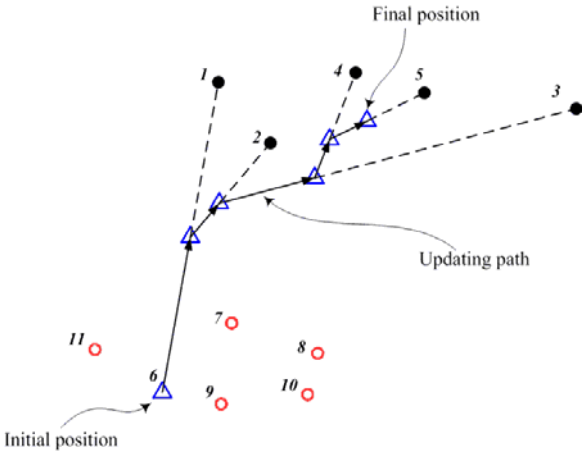

Fig. 2. Schematic representation of updating path of one firefly in the original FA. The triangle shows position of a firefly whose updating path is drawn. The solid circles are brighter fireflies and the hollow circles are the rest of them

To overcome this point, a simple updating formula is proposed to remove the wandering motion of the fireflies. In this approach, instead of moving each firefly toward the brighter ones in a stepwise manner, a representative point which shows the overall distribution of the brighter fireflies is defined at first and then the firefly moves toward this point in only one step. In other words, the updating formula for any firefly i which is attracted by a set of brighter fireflies is proposed as follows.

$$\mathbf{x}_i = \beta(\mathbf{p}_i - \mathbf{x}_i) + \alpha\boldsymbol{\varepsilon} \tag{4}$$

where $\mathbf{p}_i$ is the representative point that shows the overall distribution of the brighter fireflies. Various ideas can be invoked to define this representative point $\mathbf{p}_i$. The simplest one which is used here is to define the coordinates of the point $\mathbf{p}_i$ as the average of the coordinates of the brighter fireflies as follows.

$$\mathbf{p}_i = \frac{1}{i-1}\sum_{l=1}^{i-1}\mathbf{x}_l \tag{5}$$

A schematic representation of the above updating formula is shown in Fig. 3.

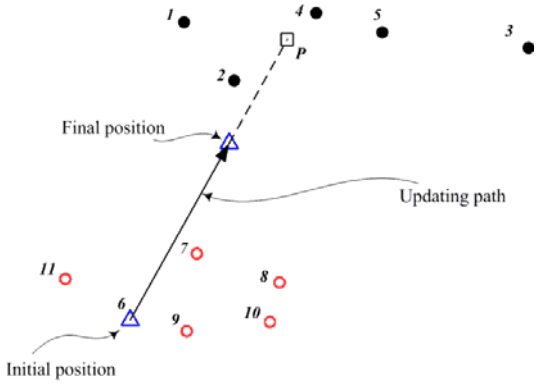

Fig. 3. Schematic representation of updating path based on the proposed updating formula. The triangle is the firefly whose updating path is drawn. The solid circles are brighter fireflies and the hollow circles are the rest of them. The square is the representative point of brighter fireflies

Based on the modifications described in the foregoing sections, a pseudo code is prepared and is shown in Fig. 4. Both of the memory approaches are considered in this pseudo code. The memory parameters m_1 and m_2 must be chosen in such a way that only one approach becomes active.

Define the upper bound **U** *and lower bound* **L** *for the design variables*
Generate an initial population of fireflies $\mathbf{x}_i$ *(i=1 to n)*
Evaluate the response function F_i *for each firefly* $\mathbf{x}_i$
Sort the fireflies based on their response function
for *t=1 to Maximum iteration*
  $\mathbf{y}_i=\mathbf{x}_i$ *(i=1 to n)*
  for $i=m_1$ *to* $n-m_2-k$
    $\mathbf{p}$*=average of coordinates of fireflies that are brighter than* $\mathbf{x}_i$
    $r=norm(\mathbf{x}_i-\mathbf{p})$
    $\beta = \beta_0 \times \exp(-\gamma r^2)$
    $\boldsymbol{\varepsilon} = (rand - 0.5) \times (\mathbf{U} - \mathbf{L})$
    $\mathbf{x}_i = \mathbf{x}_i + \beta(\mathbf{P} - \mathbf{x}_i) + \alpha\boldsymbol{\varepsilon}$
  next *i*
  Check the side constraints for firefly $\mathbf{x}_i$
  for $i=n-m_2-k+1$ *to* $n-k$
    $\mathbf{x}_i=\mathbf{y}_{i-n+m2+k}$
  next *i*
  for $i=n-k+1$ *to* n
    $\mathbf{x}_i=\mathbf{L}+rand\times(\mathbf{U}-\mathbf{L})$
  next *i*
  Evaluate the response function F_i *only for the updated fireflies*
  Sort the fireflies based on their response function
  Present the first firefly as the best solution obtained in this iteration
next *t*

Fig. 4. Pseudo code of the proposed modified firefly algorithm

4. CONSTRAINT HANDLING APPROACH

The most common approach in the metaheuristic optimization community to handle constraints is to use the penalty method. The basic idea of this method is to transform a constrained optimization problem into an unconstrained one by adding a certain value to the objective function based on the amount of constraint

violation occurred in a certain solution. Such technique, which is known as the exterior penalty method, is one of the most popular methods of constraint handling in the evolutionary algorithms. A similar method is also used in the present work.

If the optimization problem consists of minimization of cost function f subjected to the inequality constraints $g_i \le 0, (i = 1 \text{ to } p)$ and equality constraints $h_i = 0, (i = 1 \text{ to } q)$, then in the penalty function approach, the constraints can be collapsed with the cost function into a response functional F defined as follows:

$$F = f + \sum_{i=1}^{p} \lambda_i (g_i^+)^2 + \sum_{i=1}^{q} \mu_i h_i^2 \tag{6}$$

$$g_i^+ = \max(g_i, 0) \tag{7}$$

where $\lambda_i >> 0$ and $\mu_i >> 0$ are the penalty coefficients. The penalty coefficients should be large enough to obtain a feasible solution and may depend on the specific optimization problem. By doing this, the constrained optimization problem is transformed into an unconstrained optimization problem which is simpler to solve.

5. EVALUATING EXAMPLES

Most real world engineering optimization problems are nonlinear with complex objective and constraints functions. Evaluation of these functions is also time consuming and expensive. Therefore, a good algorithm, from this point of view, is one which captures the optimum point with a minimum number of function evaluations. Standard test problems are useful for the purpose of evaluating optimization algorithms. Six benchmark numerical examples considered in this section have been widely used for this purpose. In all cases, the second approach for the memory is used and the statistical measures have been obtained and reported based on 50 independent runs.

a) Welded beam design problem

In the first example, a horizontal beam which is to be welded to a vertical rigid column must be designed for minimum cost to support a tip load [33] (Fig. 5). The beam is made of C-1010 carbon steel and is under vertical load $P = 6$ kips . The thickness of the weld (h), the length of the welded joint (l), the width of the beam (t) and the thickness of the beam (b) were considered as design variables. The length L is assumed to be specified at 14 in. The optimization problem consists of selecting the design variables in such a manner to minimize the fabrication cost subject to constraints on shear stress in weld (τ), bending stress in the beam (σ), critical buckling load (P_c), tip deflection of the beam (δ) and also side constraints.

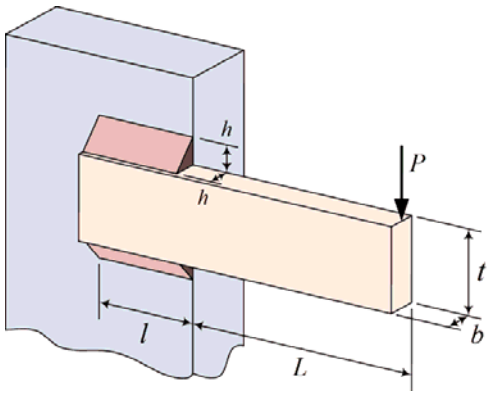

Fig. 5. Welded beam design problem

The objective function is as follows.

$$f(h,l,t,b) = 1.10471h^2l + 0.04811tb(14+l) \tag{8}$$

which is subjected to the following constraints.

$$g_1 = \tau - \tau_{all} \le 0 \tag{9}$$

$$g_2 = \sigma - \sigma_{all} \le 0 \tag{10}$$

$$g_3 = h - b \le 0 \tag{11}$$

$$g_4 = P - P_c \le 0 \tag{12}$$

$$g_5 = \delta - \delta_{all} \le 0 \tag{13}$$

The allowable shear stress, normal stress and tip deflection are $\tau_{all} = 13.6\,\text{ksi}$, $\sigma_{all} = 30\,\text{ksi}$ and $\delta_{all} = 0.25\,\text{in}$ respectively. The resultant of primary and secondary shear stresses in the weld is as follows.

$$\begin{aligned} &\tau = \sqrt{\tau'^2 + \tau''^2 + \tau'\tau''l \Big/ \sqrt{(l^2 + (h+t)^2)/4}}, \\ &\tau' = \frac{6000}{\sqrt{2}hl}, \quad \tau'' = \frac{6000(14 + l/2)\sqrt{(l^2 + (h+t)^2)/4}}{\sqrt{2}hl(l^2/12 + (h+t)^2/4)} \end{aligned} \tag{14}$$

The maximum bending stress in the root of the beam is as follows.

$$\sigma = 504000/(t^2b) \tag{15}$$

The critical buckling load is as follows.

$$P_c = 64746(1 - 0.0282346t)tb^3 \tag{16}$$

The maximum tip deflection is as follows.

$$\delta = 2.1952(t^3b) \tag{17}$$

The side constraints are:

$$0.125 \le h \le 5, \quad 0.1 \le l \le 10, \quad 0.0065 \le t \le 5, \quad 0.1 \le b \le 5, \tag{18}$$

The problem is solved using proposed method with *n=20*, $m_1 = 0$, $m_2 = 2$ and *k=1*. The proposed algorithm found the optimum solution requiring 1500 iterations per optimization run. The statistical results are obtained in the present work for 50 independent runs and are presented in Table 1. The convergence history plot for the best and average runs are also shown in Fig. 6. Table 2 compares the optimization results found in the present work with similar data reported in literature. As it can be seen the proposed method requires 27000 function evaluations to complete the optimization process.

Table 1. Statistical results for 50 independent runs for the welded beam design problem

Best	2.3822
Mean	2.5356
Worst	5.4268
Standard deviation	0.1051
No. fireflies	20
No. memories, m_2	2
No. newborn fireflies, k	1
No. iterations	1500
No. Function evaluations	27000

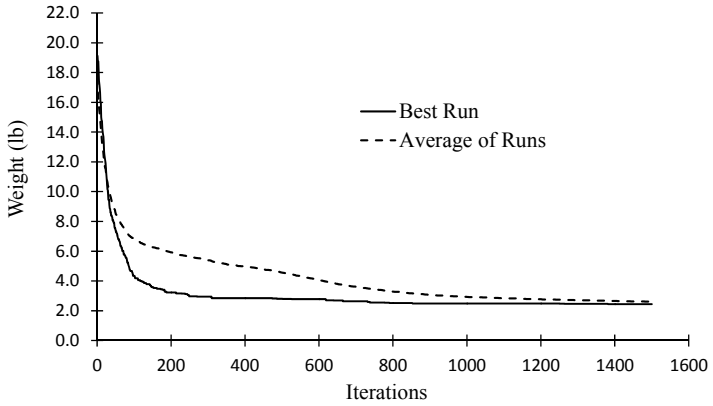

Fig. 6. Convergence history for the welded beam design problem

Table 2. Optimization results presented in different sources for welded beam design problem

Reference	h	l	t	b	Cost	no. of func. eval.
Siddall 1972 [33]	0.2444	6.2819	8.2915	0.2444	2.3815	N.A.
Ragsdell and Phillips 1976 [34]	0.2444	6.2186	8.2915	0.2444	2.3811	N.A.
Deb 1991 [35]	0.2489	6.1730	8.1789	0.2533	2.4331	320080
Leite and Topping 1998 [36]	0.2489	6.1097	8.2484	0.2485	2.4000	6273
Atiqullah and Rao 2000 [37]	0.2471	6.1451	8.2721	0.2495	2.4148	N.A.
Deb 2000 [38]	N.A.	N.A.	N.A.	N.A.	2.3819	40080
Akhtar et al. 2002 [39]	0.2407	6.4851	8.2399	0.2497	2.4426	19259
Ray and Liew 2003 [40]	0.2444	6.2380	8.2886	0.2446	2.3854	33095
He et al. 2004 [41]	0.2444	6.2175	8.2915	0.2444	2.3810	30000
Lemonge and Barbosa 2004 [42]	0.2443	6.2117	8.3015	0.2443	2.3816	320000
Lee and Geem 2005 [43]	0.2442	6.2231	8.2915	0.2443	2.3810	110000
Liu 2005 [44]	0.2444	6.2175	8.2915	0.2444	2.3810	N.A.
Hedar and Fukushima 2006 [45]	0.2444	6.2158	8.2939	0.2444	2.3811	56243
Zhang et al. 2008 [46]	0.2444	6.2175	8.2915	0.2444	2.3810	24000
Zhang et al. 2009 [47]	0.2443	6.2201	8.2940	0.2444	2.3816	28897
Aragon et al. 2010 [48]	0.2444	6.2186	8.2915	0.2444	2.3811	320000
Present study	0.2443	6.2251	8.2916	0.2444	2.3822	27000

b) Pressure vessel design problem

A cylindrical pressure vessel capped at both ends by hemispherical heads must be designed for minimum cost to endure an internal pressure [49] (Fig. 7). The vessel must be designed according to the ASME code on boilers and pressure vessels. The internal working pressure is 3 ksi and minimum volume requirement is 750 ft^3. The total cost results from a combination of welding, material and forming costs. The thickness of the cylindrical shell (T_s), the thickness of the spherical head (T_h), the inner radius (R) and the length of the cylindrical segment of the vessel (L) were considered as the design variables. The objective function can be stated as follows.

$$f(T_s, T_h, R, L) = 0.6224 T_s R L + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_h^2 R \tag{19}$$

In accordance with the ASME design codes, the constraints are stated as follows.

$$g_1 = -T_s + 0.193R \le 0 \tag{20}$$

$$g_2 = -T_h + 0.00954R \le 0 \tag{21}$$

$$g_3 = -\pi R^2 L - \frac{4}{3}\pi R^3 + 750 \times 1728 \le 0 \tag{22}$$

The side constraints are:

$$0.0625 \le T_s \le 6.1875, \quad 0.0625 \le T_h \le 6.1875, \quad 10 \le R \le 200, \quad 10 \le L \le 240 \tag{23}$$

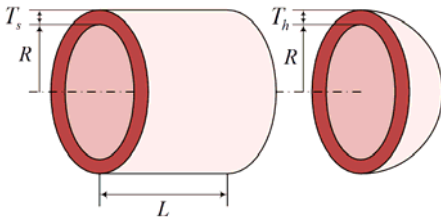

Fig. 7. Pressure vessel design problem

The proposed method is used with $n=20$, $m_1 = 0$, $m_2 = 2$ and $k=1$ and the optimization results are presented in Table 3 after 1500 iterations. The results are compared with other results given in the literature in Table 4. As it can be seen, the total number of function evaluations is 27000 per each independent run.

Table 3. Statistical results for 50 independent runs for the pressure vessel design problem

Best	6048.5142
Mean	7255.0734
Worst	9010.8501
Standard deviation	393.6952
No. fireflies	20
No. memories, m_2	2
No. newborn fireflies, k	1
No. iterations	1500
No. Function evaluations	27000

Table 4. Optimization results presented in different sources for pressure vessel design problem

Reference	Ts	Th	R	L	Cost	No. of func. eval.
Sandgren 1990 [49]	1.1250	0.6250	48.9700	106.7200	7982.5000	N.A.
Zhang and Wang 1993 [50]	1.1250	0.6250	58.2900	43.6930	7197.7000	N.A.
Deb 1997 [51]	0.9375	0.5000	48.3290	112.6790	6410.3811	N.A.
Coello 2000 [52]	0.8125	0.4375	40.3239	200.0000	6288.7445	N.A.
Coello and Montes 2002 [53]	0.8125	0.4375	42.0974	176.6540	6059.9463	80000
Hedar and Fukushima 2006 [45]	0.7683	0.3797	39.8096	207.2256	5868.7648	108883
Mahdavi et al. 2007 [54]	0.7500	0.3750	38.8601	221.3655	5849.7617	N.A.
Dimopoulos 2007 [55]	0.7500	0.3750	38.8601	221.3655	5850.3831	100000
Cagnina et al. 2008 [56]	0.8125	0.4375	42.0984	176.6366	6059.7143	24000
Shen et al. 2009 [57]	0.8125	0.4375	42.0984	176.6366	6059.7140	26000
Aragon et al. 2010 [48]	0.8125	0.4375	42.0984	190.7877	6390.5540	80000
Coelho 2010 [58]	0.8125	0.4375	42.0984	176.6372	6059.7208	8000
Gandomi et al. 2011 [59]	0.7500	0.3750	38.8601	221.3655	5850.3831	25000
Gandomi et al. 2012 [60]	0.8125	0.4375	42.0984	176.6366	6059.7100	15000
Ali 2012 [61]	0.7782	0.3846	40.3197	200.0000	5885.3300	31915
Present study	0.7902	0.3828	39.9187	206.8192	6048.5142	27000

c) Tension-compression spring design problem

The helical tension-compression spring shown in Fig. 8 is subject to an axial load. The spring must be designed for minimum weight [62].The wire diameter (d), mean coil diameter (D) and the number of

active coils (N) were included as optimization variables. The objective function can be expressed as follows.

$$f(d,D,N) = Dd^2(N+2) \tag{24}$$

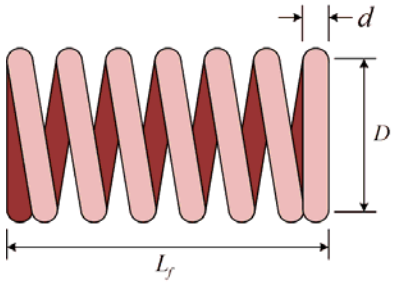

Fig. 8. Tension-compression spring design problem

The following four normalized constraints specify the design limitations:

The minimum deflection of the spring caused by the axial loading should be greater than a specified value.

$$g_1 = 1 - D^3N/(71875d^4) \le 0 \tag{25}$$

The maximum shear stress in the wire should be smaller than the allowable shear stress of the material.

$$g_2 = D(4D-d)\left(12566d^3(D-d)\right) + 2.46/(12566d^2) - 1 \le 0 \tag{26}$$

The surge frequency should be greater than a specified value.

$$g_3 = 1 - 140.54d/(D^2N) \le 0 \tag{27}$$

Finally, a limit exists on the outside diameter of the spring

$$g_4 = (D+d)/1.5 - 1 \le 0 \tag{28}$$

The side constraints are:

$$0.05 \le d \le 0.2, \quad 0.25 \le D \le 1.3, \quad 2 \le N \le 15 \tag{29}$$

The optimization results obtained by the present method are given in Table 5. With $n=15$, $m_1 = 0$, $m_2 = 2$ and $k=1$ the optimization process was completed within 1000 iterations and 13000 function evaluations. Table 6 compares the optimization results found in the current work with similar data reported in literature.

Table 5. Statistical results for 50 independent runs for the spring design problem

Best	0.01269
Mean	0.01513
Worst	0.02320
Standard deviation	0.00147
No. fireflies	15
No. memories, m_2	2
No. newborn fireflies, k	1
No. iterations	1000
No. Function evaluations	13000

Table 6. Optimization results presented in different sources for spring design problem

Reference	d	D	N	Cost	no. of func. eval.
Arora 1989 [62]	0.05340	0.39918	9.18540	0.01273	N.A.
Coello 2000 [52]	0.05148	0.35166	11.63220	0.01270	N.A.
Ray and Saini 2001 [63]	0.05042	0.32153	13.97990	0.01306	1291
Ray and Liew 2003 [40]	0.05216	0.36816	10.64840	0.01267	30000
He et al. 2004 [41]	0.05169	0.35675	11.28710	0.01267	15000
Coello and Becerra 2004 [64]	0.05000	0.31740	14.03180	0.01272	50000
Parsopoulos and Vrahatis 2005 [65]	N.A.	N.A.	N.A.	0.01312	100000
Hedar and Fukushima 2006 [45]	0.05174	0.35800	11.21390	0.01267	49531
Huang et al. 2007 [66]	0.05161	0.35471	11.41080	0.01267	204800
He and Wang 2007 [67]	0.05173	0.35764	11.24450	0.01267	200000
Hsu and Liu 2007 [68]	0.05236	0.37315	10.36490	0.01265	N.A.
Zhang et al. 2008 [46]	0.05169	0.35672	11.28900	0.01267	24000
Cagnina et al. 2008 [56]	0.05158	0.35419	11.43868	0.01267	24000
Shen et al. 2009 [57]	0.05169	0.35677	11.28617	0.01267	8000
Aragon et al. 2010 [48]	0.05162	0.35511	11.38450	0.01267	36000
Coelho 2010 [58]	0.05151	0.35253	11.53890	0.01267	2000
Gandomi et al. 2012 [60]	0.05169	0.35673	11.28850	0.01267	5000
Ali 2012 [61]	0.05184	0.36030	11.08220	0.01267	4945
Present study	0.05173	0.35770	11.25950	0.01269	13000

d) Ten bar plane truss

Figure 9 shows dimensions, loadings and supports of a ten bar plane truss structure. This structure is a standard example used by many authors to evaluate optimization algorithms. In this example, two concentrated vertical force of $P=100$ *kips* are applied on nodes 2 and 4 (Fig. 9) and all members are assumed to be made from a material with Young's modulus of $E=10000$ *ksi* and mass density of $\rho=0.10$ *lb/in*^{3}. The objective function is considered as total weight of the structure and cross sectional areas of all members are considered as the design variables. The minimum and maximum cross sectional areas of members are set to $A_{min}=0.1$ *in*^{2} and $A_{max}=35.0$ *in*^{2}. The maximum allowable displacement of free nodes in each direction and maximum allowable stress in the members are considered to be less than $\delta_{all}=2$ *in* and $\sigma_{all}=25$ *ksi*, respectively. In summary, in this example, there are 10 design variables and 36 nonlinear constraints (10 tension constraints, 10 compression constraints and 16 displacement constraints).

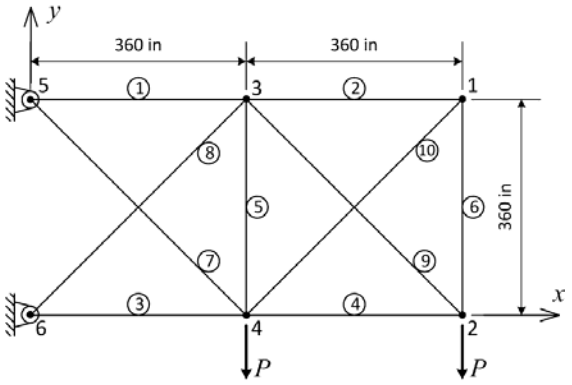

Fig. 9. Ten bar plane truss structure

In this example, and also in the following two ones, we don't have closed form representation of the constraints. Therefore, the finite element method is used here to evaluate the nodal displacements and elemental stress of the structure. Basic procedures and formulation of the finite element solution of truss structures can be found in any textbook such as [69].

The sizing optimization problem of this truss is solved using the proposed method with $n=25$, $m_1 = 0$, $m_2 = 2$ and $k=1$. After 2500 iterations, in each independent run the proposed algorithm found the optimum solution. The results obtained in the present work as well other results which appeare in the literature are presented in Table 7.

Table 7. Optimal cross sectional areas (in^2) for ten bar plane truss problem

Element numbers	Hatay and Toklu 2002 [71]	Li et al. 2007 [72]	Sonmez 2011 [73]	Present study
1	30.680	30.704	30.548	29.808
2	0.100	0.100	0.100	0.100
3	23.500	23.167	23.180	23.080
4	14.970	15.183	15.218	15.288
5	0.100	0.100	0.100	0.100
6	0.550	0.551	0.551	0.578
7	7.450	7.460	7.463	7.467
8	21.020	20.978	21.058	21.296
9	21.430	21.508	21.501	21.800
10	0.100	0.100	0.100	0.100
Weight (*lb*)	5061.600	5060.92	5060.880	5060.890

e) Eighteen bar plane truss

Figure 10 shows the geometry and loading condition of the cantilever truss structure consisting of eighteen members. The structure is subjected to a series of concentrated point forces of $P=20$ *kips* acting on the upper cord nodes of the truss as shown in Fig. 10. All members are made from material with an elastic modulus of $E=10000$ *ksi* and a mass density of $\rho=0.10$ *lb/in*^{3}. The stress constraint is defined as $\sigma_{all}=20$ *ksi* for both the tension and compression members. In addition, the linear (Euler) buckling constraint is also taken into account for compression members. The critical buckling stress for the *i th* member is obtained as:

$$\sigma_i = -KEA_i /(L_i^2) \tag{30}$$

where L_i is length of the member, A_i is cross sectional area of the member and K is a constant determined from geometry and was taken to be 4. The number of independent size variables was reduced to four groups as presented in Table 8. The minimum cross sectional area of the members is $A_{min}=0.10\ in^2$ and the maximum cross section is set to $A_{max}=50\ in^2$. There are 36 nonlinear constraints on the member stress and buckling stress with no displacement constraints.

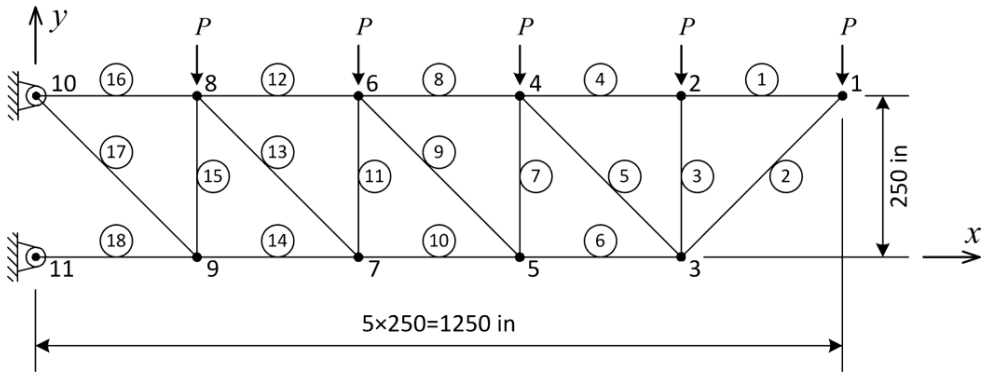

Fig. 10. Eighteen bar plane truss structure

Table 8. Optimal cross sectional areas (in^2) for eighteen bar plane truss problem

Element numbers	Imai and Schmit 1991 [74]	Lee and Geem 2004 [75]	Sonmez 2011 [73]	Present study
1, 4, 8, 12, 16	9.998	9.980	10.000	10.000
2, 6, 10, 14, 18	21.650	21.630	21.651	21.650
3, 7, 11, 15	12.500	12.490	12.500	12.500
5, 9, 13, 17	7.072	7.057	7.071	7.071
Weight (*lb*)	6430.000	6421.880	6430.529	6430.433

Displacements and stresses are obtained using the finite element method. The sizing optimization problem of this truss is solved using the proposed method with $n=25$, $m_1 = 0$, $m_2 = 2$ and $k=1$. After 2500 iterations, the results are presented in Table 8. In addition, this table contains the results for the same optimization task from different research efforts.

f) Twenty five bar space truss

As the last numerical example, a spatial truss structure that has been solved by many researchers as a benchmark structural problem is selected [70]. In this example, the weight minimization of a 25 bar transmission tower structure under two different loading cases is considered. Its topology and node numbers are shown in Fig. 11. The design variables are the cross sectional area of the truss members which are categorized in eight groups A to H as shown in Fig. 11. The structure is subjected to two loading conditions as presented in Table 9. The design constraints are the maximum displacement limitations of $\delta_{all}=0.35\ in$ on every node in every direction and the axial stresses which are shown in Table 10 for each group of the truss members. The range of cross sectional area of the truss members is selected from $A_{min}=0.10\ in^2$ to $A_{max}=3.4\ in^2$. The material density is considered as $\rho=0.10\ lb/in^3$ and the modulus of elasticity is taken as $E=10000\ ksi$.

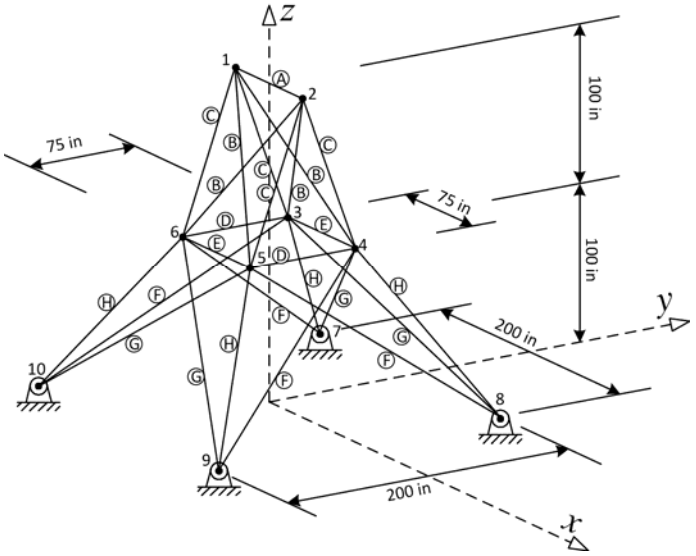

Fig. 11. Twenty five bar space truss structure

Table 9. Loading cases for 25 bar space truss problem

Load case	Node	P_x (*lb*)	P_y (*lb*)	P_z (*lb*)
1	1	1.0	10.0	-5.0
	2	0.0	10.0	-5.0
	3	5.0	0.0	0.0
2	5	0.0	20.0	-5.0
	6	0.0	-20.0	-5.0

Table 10. Stress limitations for each element group for 25 bar space truss problem

Element group	Tensile stress limitations (*ksi*)	Compressive stress limitations (*ksi*)
A	40.0	35.092
B	40.0	11.590
C	40.0	17.305
D	40.0	35.092
E	40.0	35.092
F	40.0	6.759
G	40.0	6.959
H	40.0	11.082

Similar to previous truss examples, the finite element method is used to determine the stresses and displacements. The weight optimization problem of this space truss is also solved using the proposed method with $n=25$, $m_1 = 0$, $m_2 = 2$ and $k=1$. After 3000 iterations, the results are presented in Table 11. This table also illustrates the optimum results reported in the literature.

Table 11. Optimal cross sectional areas (in^2) for 25 bar space truss problem

Element group	Haftka and Gürdal 1992 [76]	Lee and Geem 2004 [75]	Li et al. 2007 [72]	Lamberti 2008 [77]	Sonmez 2011 [73]	Present study
A	0.010	0.047	0.010	0.010	0.011	0.010
B	1.987	2.022	1.970	1.987	1.979	1.997
C	2.991	2.950	3.016	2.994	3.003	3.011
D	0.010	0.010	0.010	0.010	0.010	0.010
E	0.012	0.014	0.010	0.010	0.010	0.010
F	0.683	0.688	0.694	0.694	0.690	0.687
G	1.679	1.657	1.681	1.681	1.679	1.644
H	2.664	2.663	2.643	2.643	2.652	2.678
Weight (*lb*)	545.220	544.380	545.190	545.161	545.193	545.114

6. CONCLUSION

In the present article, three basic modifications were proposed to improve performance of the firefly optimization algorithm. They were: adding memory, adding mutation and defining a new updating formula. The added memory stores valuable information in each iteration and transfers it to the next iteration. The mutation promotes diversification of the optimizer in searching of the entire solution space for potential optima. The proposed updating formula overcomes wandering motion of the fireflies. Six numerical examples consistsing of three classical engineering optimization problems as well as three sizing optimization of truss structures were selected as the benchmark problems to evaluate performance of these modifications. It is empirically observed that adding memory can significantly improve the performance of the algorithm. Nevertheless, the second memory approach can be trapped in local optima. It is also seen that this point can be repaired using the mutation. The mutation is also useful when the initial population is distributed oddly over the search space. It is also observed that the proposed updating formula removes the zigzag motion of the fireflies in the updating process. Based on these observations, it is believed that the proposed method can be efficiently used in real life engineering optimization problems.

REFERENCES

1. Kazemzadeh-Parsi, M. J. & Daneshmand, F. (2013). Inverse geometry heat conduction analysis of functionally graded materials using smoothed fixed grid finite elements. *Inverse Problems in Science and Engineering*, Vol. 21, No. 2, pp. 235-250.

2. Talbi, E. (2009). *Metaheuristics: from design to implementation*. Hoboken, New Jersey, USA: John Wiley & Sons.
3. Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms*. 2nd edition, United Kingdom: Luniver Press.
4. Goldberg, D. (1989). *Genetic algorithms in search optimization and machine learning*. Addison-Wesley.
5. Back, T., Hoffmeister, F. & Schwefel, H. (1991). A survey of evolution strategies. *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*, 2-9.
6. Fogel, L. (1994). *Evolutionary programming in perspective: The top-down view*. In: Computational Intelligence: Imitating Life, edited by Zurada, J.M., Marks, Jr., R., and Robinson, C., IEEE Press, Piscataway, NJ, USA.
7. Eberhart, R. C. & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings of the sixth international symposium on micro machine and human science*, Nagoya, Japan.
8. Storn, R. & Price, K. (1995). Differential evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, USA.
9. Dorigo, M., Maniezzo, V. & Colorni, A. (1996). The ant system: optimization by a colony of cooperating agents. *IEEE Trans Syst Man Cybern B*, Vol. 26, No. 1, pp. 29-41.
10. Nakrani, S. & Tovey, C. (2004). On honey bees and dynamic server allocation in Internet hostubg centers. *Adaptive Behavior*, Vol. 12, pp. 223-240.
11. Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. & Zaidi, M. (2005). *The bees algorithm*. Technical Note, Manufacturing Engineering Center, Cardiff University.
12. Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical Report, Erciyes University.
13. Yang, X. S. & Deb, S. (2009). Cuckoo search via Lévy flights. *Proceedings of the world congress on nature & biologically inspired computing (NaBIC 2009)*, IEEE Publications, pp. 210-214.
14. Oftadeh, R., Mahjoob, M. J. & Shariatpanahi, M. (2010). A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Comput Math Appl*, Vol. 60, pp. 2087-2098.
15. Yang, X. S. (2010b). A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization NISCO edited by Gonzalez, J.R. et al., *Studies In Computational Intelligence*, Vol. 284. Berlin: Springer, pp. 65-74.
16. Kirtrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, Vol. 220, (4598), pp. 671-680.
17. Geem, Z. W., Kim, J. H. & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony Search. *Simulation*, Vol. 76, No. 2, pp. 60-68.
18. Osman, K. & Erol, I. E. (2006). A new optimization method: Big Bang-Big Crunch. *Adv Eng Software*, Vol. 37, No. 2, pp. 106-111.
19. Kaveh, A. & Talatahari, S. (2010). A novel heuristic optimization method: charged system search. *Acta Mech*, Vol. 213, pp. 267-289.
20. Tamura, K. & Yasuda, K. (2011). Spiral dynamics inspired optimization. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 15, No. 8, pp. 1116-1122.
21. Boussaid, I., Chatterjee, A., Siarry, P. & Ahmed-Nacer, M. (2012). Biogeography-based optimization for constrained optimization problems. *Computers & Operations Research*, Vol. 39, pp. 3293-3304.
22. Gandomi, A. H. & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Commun Nonlinear Sci Numer Simulat*, Vol. 17, pp. 4831-4845.
23. Rao, R. V., Savsani, V. J. & Balic, J. (2012). Teaching–learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems. *Engineering Optimization*, DOI:10.1080/0305215X.2011.652103.
24. Kaveh, A. & Khayatazad, M. (2012). A new meta-heuristic method: Ray optimization. *Computers and Structures*, Vol. 112-113, pp. 283-294.

25. Fister, I., Fister Jr., I., Yang, X. S. & Brest, J. (2013). A comprehensive review of firefly algorithms:, Swarm and Evolutionary Computation, http://dx.doi.org/10.1016/j.swevo.2013.06.001
26. Kazemzadeh-Parsia, M. J., Daneshmand, F., Ahmadfard, M. A., Adamowski, J. & Martel, R. (2013). Optimal groundwater remediation design of pump and treat systems via a simulation-optimization approach and firefly algorithm. *Engineering Optimization*, DOI:10.1080/0305215X.2013.858138.
27. Yang, X. S. (2009). Firefly algorithms for multimodal optimization. in: Stochastic Algorithms: Foundations and Applications, SAGA. *Lecture Notes in Computer Sciences*, Vol. 5792, pp. 169-178.
28. Wang, G., Guo, L., Duan, H., Liu, L. & Wang, H. (2012). A modified firefly algorithm for UCAV path planning. *International Journal of Hybrid Information Technology*, Vol. 5, No. 3.
29. Coelho, L. D. S. & Mariani, V. C. (2012). Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. *Computers & Mathematics with Applications*, Vol. 64, No. 8, pp. 2371-2382.
30. Gandomi, A. H., Yang, X. S., Talatahari, S. & Alavi, A. H. (2013). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, Vol. 18, No. 1, pp. 89-98.
31. Yang, X. S. (2013). Multi-objective firefly algorithm for continuous optimization. *Engineering with Computers*, Vol. 29, No. 2, pp. 175-184.
32. Ahmadfard, M. A., Kazemzadeh-Parsi, M. J., Daneshmand, F. & Adamowski, J. (2014). Optimal remediation design of unconfined contaminated aquifers based on the finite element method and a modified version of the firefly algorithm. Submitted to Engineering Optimization.
33. Siddall, J. N. (1972). *Analytical decision making in engineering design*. Englewood Cliffs: Prentice-Hall.
34. Ragsdell, K. M. & Phillips, D. T. (1976). Optimal design of a class of welded structures using geometric programming. *ASME J Eng Ind*, Vol. 98, No. 3, pp. 1021-1025.
35. Deb, K. (1991). Optimal design of a welded beam via genetic algorithms. *AIAA J*, Vol. 29, No. 11, 2013–2015.
36. Leite, J. P. B. & Topping, B. H. V. (1998). Improved genetic operators for structural optimization" Adv Eng Software. Vol. 29, No. 7-9, pp. 529-562.
37. Atiqullah, M. M. & Rao, S. S. (2000). Simulated annealing and parallel processing: an implementation for constrained global design optimization. *Eng Optimiz*, Vol. 32, No. 5, pp. 659-685.
38. Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Comput Methods Appl Mech Eng*, Vol. 186, pp. 311-338.
39. Akhtar, S., Tai, K. & Ray, T. (2002). A socio-behavioural simulation model for engineering design optimization. *Eng Optimiz*, Vol. 34, No. 4, pp. 341-354.
40. Ray, T. & Liew, K. M. (2003). Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Trans Evolution Comput*, Vol. 7, No. 4, pp. 386-396.
41. He, S., Prempain, E. & Wu, Q. H. (2004). An improved particle swarm optimizer for mechanical design optimization problems. *Eng Optimiz*, Vol. 36, No. 5, pp. 585-605.
42. Lemonge, A.C.C. & Barbosa, H.J.C. (2004). An adaptive penalty scheme for genetic algorithms in structural optimization. *Int J Numer Methods Eng*, Vol. 59, pp. 703-736.
43. Lee, K. S. & Geem, Z. W. (2005). A new metaheuristic algorithm for continues engineering optimization: harmony search theory and practice. *Comput Methods Appl Mech Eng*, Vol. 194, pp. 3902-3933.
44. Liu, J. L. (2005). Novel orthogonal simulated annealing with fractional factorial analysis to solve global optimization problems. *Eng Optimiz*, Vol. 37, No. 5, pp. 499-519.
45. Hedar, A. R. & Fukushima, M. (2006). Derivative-free filter simulated annealing method for constrained continuous global optimization. *J Global Optimiz*, Vol. 35, 4, pp. 521-649.
46. Zhang, M., Luo, W. & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Inform Sci*, Vol. 178, No. 15, pp. 3043-3074.

47. Zhang, J., Liang, C., Huang, Y., Wu, J. & Yang, S. (2009). An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. *Appl Math Comput*, Vol. 211, pp. 392-416.
48. Aragon, V. S., Esquivel, S. C. & Coello, C.A.C. (2010). A modified version of a T-Cell Algorithm for constrained optimization problems. *Int. J. Numer. Meth. Engng*, Vol. 84, pp. 351-378.
49. Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *ASME J Mech Des*, Vol. 112, No. 2, pp. 223-229.
50. Zhang, C. & Wang, H. P. (1993). Mixed-discrete nonlinear optimization with simulated annealing. *Engineering Optimization*, Vol. 17, No. 3, pp. 263-280.
51. Deb, K. (1997). *GeneAS: A robust optimal design technique for mechanical component design*. in Evolutionary Algorithms in Engineering Applications, edited by Dasgupta, D., and Michalewicz, Z., Berlin, Springer-Verlage, pp. 497-514.
52. Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Comput Indus*, Vol. 41, No. 2, pp. 113-127.
53. Coello, C. A. C. & Montes, E. F. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv. Eng. Inf*., Vol. 16, pp. 193-203.
54. Mahdavi, M., Fesanghary, M. & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Appl Math Comput*, Vol. 188, pp. 1567-1579.
55. Dimopoulos, G. G. (2007). Mixed-variable engineering optimization based on evolutionary and social metaphors. *Comput. Methods Appl. Mech. Engrg*, Vol. 196, pp. 803-817.
56. Cagnina, L.C., Esquivel, S.C. & Coello, C.A.C. (2008). Solving engineering optimization problems with the simple constrained particle optimizer. *Informatica*, Vol. 32, pp. 319-26.
57. Shen, H., Zhu, Y., Niu, B. & Wu, Q. H. (2009). An improved group search optimizer for mechanical design optimization problems. *Progress in Natural Science*, Vol. 19, pp. 91-97.
58. Coelho, L.D.S. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications*, Vol. 37, pp. 1676-1683.
59. Gandomi, A. H., Yang, X. S. & Alavi, A. H. (2011). Mixed variable structural optimization using Firefly Algorithm. *Computers and Structures*, Vol. 89, pp. 2325-2336.
60. Gandomi, A. H., Yang, X. S., Alavi, A. H. & Talatahari, S. (2012). Bat algorithm for constrained optimization tasks. *Neural Comput & Applic*.
61. Ali, M. (2012). Swarm directions embedded differential evolution for faster convergence of global optimization problems. *international journal on artificial intelligence tools*, Vol. 21, No. 3.
62. Arora, J. S. (1989). *Introduction to Optimum Design*. New York: McGrow-Hill.
63. Ray, T. & Saini, P. (2001). Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Eng Optim*, Vol. 33, No. 3, pp. 735-748.
64. Coello, C. A. C. & Becerra, R. L. (2004). Efficient evolutionary optimization through the use of a cultural algorithm. *Eng Optim*, Vol. 36, pp. 219-236.
65. Parsopoulos, K. E. & Vrahatis, M. N. (2005). Unified particle swarm optimization for solving constrained engineering optimization problems. *Lecture Notes in Computer Science*, 3612 (LNFA), pp. 582-591.
66. Huang, F. Z., Wang, L. & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Appl. Math. Comput*., Vol. 186, pp. 340-356.
67. He, Q. & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intell*., Vol. 20, pp. 89-99.
68. Hsu, Y. L. & Liu, T. C. (2007). Developing a fuzzy proportional-derivative controller optimization engine for engineering design opti-mization problems. *Eng Optim*, Vol. 39, No. 6, pp. 679-700.
69. Rao, S. S. (2011). *The finite element method in engineering*. Fifth edition, Elsevier, New York.

70. Kaveh, A. & Bakhshpoori, T. (2013). Optimum design of space trusses using cuckoo search algorithm with lévy flights. *Iranian Journal Science and Technology, Transactions of Civil Engineering*, Vol. 37, No. 1, pp. 1-15.
71. Hatay, T. & Toklu, Y. C. (2002). Optimization of truss using simulated annealing method. *Fifth International Congress on Advances in Civil Engineering*, Istanbul Technical University, Istanbul, Turkey, 25-27 September, pp. 379-388.
72. Li, L. J., Huang, Z. B., Liu, F. & Wu, Q. H. (2007). A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers and Structures*, Vol. 85, pp. 340-349.
73. Sonmez, M. (2011). Artificial bee colony algorithm for optimization of truss structures. *Applied Soft Computing*, Vol. 11, pp. 2406-2418.
74. Imai, K. & Schmit, L.A. (1991). Configuration optimization of trusses. *Journal of Structural Division, ASCE*, Vol. 107 (ST5), pp.745-756.
75. Lee, K. S. & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers and Structures*, Vol. 82, pp. 781-798.
76. Haftka, R. T. & Gürdal, Z. (1992). *Elements of structural optimization*. 3rd ed., Kluwer Academic Publishers, The Netherlands, pp. 245–250.
77. Lamberti, L. (2008). An efficient simulated annealing algorithm for design optimization of truss structures. *Computers and Structures*, Vol. 86, pp. 1936–1953.