Structural Engineering and Mechanics, Vol. 44, No. 4 (2012) 501-519
DOI: http://dx.doi.org/10.12989/sem.2012.44.4.501

Discrete optimization of trusses using an artificial bee colony (ABC) algorithm and the fly-back mechanism

A.R. Fiouz[1a], M. Obeydi[1b], H. Forouzani[2c] and A. Keshavarz*[1]

[1]*Civil Engineering Group, Faculty of Engineering, Persian Gulf University, Bushehr, Iran*
[2]*Civil Engineering and Applied Mechanics Department, California State University, Northridge, CA 91330-8347, USA*

(Received March 8, 2012, Revised October 24, 2012, Accepted October 27, 2012)

Abstract. Truss weight is one of the most important factors in the cost of construction that should be reduced. Different methods have been proposed to optimize the weight of trusses. The artificial bee colony algorithm has been proposed recently. This algorithm selects the lightest section from a list of available profiles that satisfy the existing provisions in the design codes and specifications. An important issue in optimization algorithms is how to impose constraints. In this paper, the artificial bee colony algorithm is used for the discrete optimization of trusses. The fly-back mechanism is chosen to impose constraints. Finally, with some basic examples that have been introduced in similar articles, the performance of this algorithm is tested using the fly-back mechanism. The results indicate that the rate of convergence and the accuracy are optimized in comparison with other methods.

Keywords: discrete optimum; optimization; truss; artificial bee colony; fly-back mechanism

1. Introduction

The cost of materials is one of the key factors to consider in the construction of a building. This can be reduced by minimizing the weight or volume of the materials used. Different methods are used to minimize the volume or weight to achieve an optimal design. These methods include a set of design variables that are subjected to design constraints. Optimizing the cross-sectional area leads to a reduction of the mass and size of the structure. In addition, the optimal design must satisfy the design constraints, which limits the values of the design variables and the structure's response.

Because the use of classic mathematical methods in optimization problems can be practically difficult or impossible due to their nonlinear and nondifferentiable nature, researchers have devised methods inspired by nature to solve such problems. The evolutionary algorithm is an example of this approach.

*Corresponding author, Assistant Professor, E-mail: keshavarz@pgu.ac.ir
[a]Assistant Professor, E-mail: fiouz@pgu.ac.ir
[b]Graduate Student, E-mail: m_obeydi83@yahoo.com
[c]Graduate Student, E-mail: hani.forouzani.62@my.csun.edu

General types of algorithms for optimization problems fall into two categories (Consoli 2006):

a. Complete algorithms (exact)
b. Approximate algorithms

The methods based on swarm intelligence are a subcategory of metaheuristic methods for approximate algorithms. In these methods, in addition to considering individual performance, members of the population have social interactions with each individual member in the population. Therefore, their operation is much more effective than when acting separately.

In computational applications, organisms such as ants, bees, fish, and birds can be modeled. In general, there are several types of bee algorithms, such as the Honey Bee Algorithm (HBA), the Virtual Bee Algorithm (VBA), the Artificial Bee Colony (ABC) algorithm, and the Honey Bee Mating Algorithm (HBMA). It appears that the HBA was first formulated in approximately 2004 by Tovey at Georgia Tech in collaboration with Nakrani, then at Oxford University, to study a method to allocate computers among different clients and web-hosting servers. Later in 2004 and early 2005, Yang at Cambridge University developed a VBA to solve numerical optimization problems. In addition to solving two parameter functions, this algorithm can be used to optimize functional and discrete problems. Later in 2005, Haddad *et al*. presented a HBMO algorithm, which was subsequently applied to cluster modeling of a reservoir (Yang 2008).

Karaboga (2005) presented the idea of solving numerical problems based on bees' social behavior with an artificial bee colony. Karaboga and Basturk (2007) used this idea to present an artificial bee colony algorithm as an efficient and applicable algorithm for optimizing numerical problems and compared the results with other optimization algorithms.

Lemmens *et al*. (2007) used a bee colony to produce a new computational algorithm for multiagent systems with bees' food-finding behavior. This is different than ant colony behavior, which uses pheromones for routing, which decreases the time required to reach a given destination.

Using Newton's law of universal gravitation to make changes in the method of selecting onlooker bees, Tsai *et al*. (2009) designed an algorithm that improved numerical optimization problems (IABC).

Karaboga and Akay (2009) compared the artificial bee colony algorithm with other optimization algorithms, such as the genetic algorithm, particle swarm optimization, and evolutionary strategy. For this algorithm with fewer control parameters, they showed that the results are better than or similar to other algorithms. They (Akay and Karaboga 2012) also modified the ABC algorithm for numerical function optimization. One of the changes made is the production of a modification rate (MR) control parameter. This modification means that a random number is generated uniformly for each parameter x_{ij}, and if this number is smaller than MR, the parameter is changed. Another improvement is the scaling factor (SF) control parameter, which is related to the ratio of the variance operator in the original ABC algorithm.

Although the ABC algorithm is a powerful algorithm for discovering food sources, it performs poorly in the exploitation phase, so Zhu and Kwong (2010), inspired by PSO, developed the ABC algorithm further by adding a new term to the solution search equation.

Luo *et al*. (2010) divided ABC algorithm's particles into several independent subpopulations to increase accuracy of optimal solution.

Hadidi *et al*. (2010) improved the ABC algorithm to optimize the size of plane and space truss structure profiles by applying the concept of probability to modify neighborhood search methods and by modifying the onlooker and scout bee phase.

Kang *et al*. (2010) introduced a new algorithm based on combining an ABC algorithm with the HJ algorithm, which Hooke and Jeeves (1961) suggested as a simple search method. The new algorithm was more powerful for solving numerical problems in terms of accuracy and convergence speed than the ABC algorithm.

At first, the bee colony algorithm was introduced for unconstrained problems (Karaboga and Basturk 2007). Karaboga and Akay (2011) expanded it for constrained optimization problems by applying a scout production period control parameter to their modified algorithm (Akay and Karaboga 2012), and they used Deb's rule for constraint-handling, which was initially presented by Deb (2000) as an effective technique in genetic algorithms.

Brajevic *et al*. (2011) presented the simplified artificial bee colony algorithm (SC-ABC) with the modifications in Karaboga's algorithm (Karaboga and Akay 2011). In this algorithm during the first run, the initial population is generated randomly, and in the next runs, if there are any, the new food source in the initial population is the best answer from the previous run. Therefore, different runs are related to each other. However, the scout bee's phase will be checked for feasible solutions, and if the solution is not in a feasible space, it should be replaced with a random solution. This modification increases exploration in the scout bee's phase and the exploitation of the best food sources.

Sonmez (2011a, b) used the ABC algorithm for the first time for the discrete and continuous weight optimization of truss structures in building design.

The purpose of this study to use a combination of the ABC algorithms offered by Karaboga and Akay (2011) and Sonmez (2011a, b) to find the optimum weight of truss structures and to choose the best and lightest profiles from the list of available profiles to satisfy the design constraints and determine whether changing the constraint-handling technique could improve the convergence speed and accuracy of the ABC algorithm and lead to the optimum solution. Currently Deb's rule and penalty functions are applied to constraint-handling in the ABC algorithm, but this paper uses the fly-back mechanism technique as an improved method. Kaveh and Talatahari (2007, 2008, 2009) used this fly-back mechanism technique as a simpler method than the other techniques in their proposed algorithm (PSACO) for the optimization of trusses and steel frames. They also applied this technique in their other algorithm (HPSACO).

The content of this paper is arranged as follows.

Section 2 presents the formulation of the structural optimization problems. Section 3 describes honeybees' food-finding behavior. In Section 4, the algorithm is modeled based on the natural behavior of honeybees. In Section 5, the pseudo-code of the ABC algorithm is presented. In Section 6, the efficiency of this algorithm is tested through several standard test problems. Section 7 presents the conclusions.

2. Formulation of the structural optimization problems

The goal of optimization is to minimize or maximize the constrained functions. These functions are called objective functions. In the structures under consideration, the objective functions involve the structural weight (Togan *et al*. 2011, Yun *et al*. 2006, Kim *et al*. 2004), cost (Cagatay *et al*. 2003), shape (Lee and Park 2011), and topology (Lee and Park 2011, Chen and Rajan 2000). In this study, the objective function is the structural weight, which leads to cost reduction

$$Minimize\ W(\{A_j\}) = \sum_{j=1}^{n} A_j L_j \gamma_j \tag{1}$$

where
$W(\{A_j\})$ = Structural weight
A_j = Cross-sectional area of the j-th member
n = Number of members
L_j = Length of the j-th member
γ_j= Unit weight of the j-th member
$\{A_j\}$ = Represents a group of sections selected from a permissible profile list

$$\text{Profile list} = \{A_1, A_2, \ldots, A_z\} \tag{2}$$

where
z = Number of profiles in the list

Subjected to the following constraints

$$\delta_{\min} \le \delta_j \le \delta_{\max} \quad j = 1, 2, \ldots, m \quad \text{Displacement constraints} \tag{3}$$

$$\sigma_{\min} \le \sigma_j \le \sigma_{\max} \quad j = 1, 2, \ldots, n \quad \text{Stress constraints} \tag{4}$$

$$\sigma_j^b \le \sigma_j \le 0 \quad j = 1, 2, \ldots, nc \quad \text{Buckling stress constraints} \tag{5}$$

$$A_{\min} \le A_j \le A_{\max} \quad j = 1, 2, \ldots, ng \quad \text{Restriction of profile size} \tag{6}$$

where
m = Number of degrees of freedom
n = Number of structural members
nc = Number of compression members
ng = Number of design variables or groups
min= Lower bound
max = Upper bound
σ_j = Stress in the j-th member
δ_j = Displacement of the j-th degree of freedom
σ_j^b = Allowable buckling stress in the j-th compression member

3. Honeybees' natural food-finding behavior

Honeybees are social insects that live in colonies in hives. Each hive contains a queen, thousands of semi-sterile female workers and thousands of males (drones) (Sonmez 2011a). The male bees' only task is mating with the queen bee, whereas the workers' tasks are hive cleaning, larvae nursing, food preparation, and nest construction. The bees' food-finding process is described as follows.

There are many food sources around the hive. A number of bees are scattered around their hive looking for quality food sources. After selecting the desired flowers, they recognize the flower position using the angle of the sun and return to the hive (Sonmez 2011a). Traveling bees inform others about the sources and the quality of food through "waggle dances (Karaboga 2005). The informed bees fly toward the food source using the duration, the length of the straight path and the angle of the dance to recognize the quality, distance, and direction of the flower, respectively. They also explore the flowers' surrounding area for additional food sources (flowers with better nectar). After comparing, they choose the best source and return to the hive. By repeating this process, the best flowers are chosen.

4. An ABC modeling algorithm based on bees' natural behavior

The ABC algorithm is based on the nature of bees' behavior when searching for food. This algorithm moves toward the optimal solution by a mechanism based on selection. The artificial bee algorithm is composed of three groups (Sonmez 2011b):

1. Scout bees

The scout bees are constantly searching to discover new food sources. Initially, bees fly out into the environment as scout bees, but after finding their desired flowers, they become employed bees.

2. Employed bees

The employed bees inform other bees about the amount of nectar and the position of the flowers after returning to the hive. The information is transferred through a circular dance called the waggle dance (Karaboga 2005). The waggle dance consists of a series of circular motions. The direction of the dance is in line with the positions of the flower and the sun. Longer and better dance quality indicates better flower and nectar quality.

3. Onlooker bees

Onlooker bees remain and watch the dancing in the hive. Onlooker bees review the obtained information from employed bees in the hive, and they make selections based on the quality of the dancing.

At first, the initial space is generated randomly. This initial space represents the surrounding space around the hive, which includes *NP* (the number of desired solutions), different solutions for this problem, which is calculated using the following equation

$$A_{ij} = A_j^{\min} + \lambda_{ij}(A_j^{\max} - A_j^{\min}) \tag{7}$$

$i = 1, 2, 3, \ldots, NP$

$j = 1, 2, 3, \ldots, D$

where

A_{ij} = j-th variable of the i-th desired solution

λ_{ij} = A random number between 0 and 1

$A_j^{\max}$ = Upper bound of the j-th variable

$A_j^{\min}$ = Lower bound of the j-th variable

NP = Number of desired solutions

D = Number of design variables

Because bees search around their hive up to a certain distance (Sonmez 2011a), the initial space should be limited. This limitation is in fact the upper and lower bound of the variables in each solution. The initial space should be selected from the limited values. The ABC algorithm is essentially an algorithm based on continuous values, and it is not constrained. Bees do not have any restriction in choosing flower locations; therefore, any value could be substituted for the position of the flowers. However, in this study, due to the constrained problem and the discrete variable values, a constraint-handling technique should be used, and the obtained variable values can be substituted with the nearest available profile values. Until now, different techniques have been used for constraint-handling for optimization problems; these techniques include penalty functions, Deb's rules, and the fly-back mechanism technique. The use of penalty functions is difficult due to the difficulty in finding suitable penalty coefficients and in finding the balance between the penalty functions and the objective functions (Geem 2010). In this study, the fly-back mechanism technique is used for constraint-handling. Because the constraints of building design codes and specifications are not very complex, generating an initial feasible space is not difficult or time consuming; therefore, this technique can easily be used in structural problems.

In this technique, the global optimal situation is inside or on the boundary of the feasible region (all of the solutions in the feasible region satisfy the existing constraints). Particles (bees) are evaluated in the feasible region. When the optimization process begins, the particles (bees) search for a solution in the feasible space. If a particle (bee) moves out of the feasible region, it is returned to its previous position (Geem 2010). In fact, after generating an initial feasible population, to keep the solution feasible when a new solution is created in the neighborhood of the previous solution and if the new solution is of better quality and satisfies the constraints, the new solution will replace the previous solution. However, if the new solution does not satisfy the constraints, it will be returned to the previous solution. Because the initial population is feasible, constraint satisfaction is guaranteed. Because the initial feasible population is generated after the initial space is generated, it should be evaluated for any desired solution that satisfies the constraints. If at least one of the constraints is not satisfied, it will be replaced with a new solution. Comparing this method to other methods of constraint-handling indicates that this technique is not difficult. Some experiments have shown that this technique can find a better solution with fewer iterations (Geem 2010).

Moreover, applying the discretization method presented by Kaveh and Talatahari (2007) is useful due to its simplicity and efficiency in designing structures. They improved their proposed method with Eq. (8), which presents the particle positions in the PSO algorithm with Eq. (9).

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{8}$$

$$X_i^{k+1} = Fix(X_i^k + V_i^{k+1}) \tag{9}$$

Fix(x) is a function in Eq. (9) that fixes the value of the member of the x vector to the nearest permissible discrete value. With this function and Eq. (10), all of the values of Eq. (7) are fixed to the nearest permissible discrete value in the list of profiles.

$$A_{ij} = Fix(A_j^{\min} + \lambda_{ij}(A_j^{\max} - A_j^{\min})) \tag{10}$$

After producing the initial feasible space, all of the desired solutions are compared with existing solutions in their neighborhood (Existing solutions in the neighborhood are obtained by replacing

the value of some variables with new values)

$$A_{ij}^{new} = \begin{cases} A_{ij} + \phi_{ij}(A_{ij} - A_{kj}), & \text{if} \quad R_j < MR \\ A_{ij} & \text{otherwise} \end{cases} \tag{11}$$

However, if no variables are changed in Eq. (11), one of them is randomly chosen and replaced with the value of the neighborhood using the following equation

$$A_{ij}^{new} = A_{ij} + \phi_{ij}(A_{ij} - A_{kj}) \tag{12}$$

$i = 1, 2, 3, \ldots, NP$

$j = 1, 2, 3, \ldots, D$

$k = 1, 2, 3, \ldots, NP$

where

ϕ_{ij} = A random number between 1and −1.
R_j = A random number between 0 and 1.
k = An index that is randomly selected and should be different from i
MR = Modification rate control parameter

The result of Eqs. (11) or (12) is fixed to the nearest permissible discrete value in the list of profiles again.

The above comparison duplicates the employed bee's flower selection procedure. In this regard, any bee will select the solution that has better fitness (which represents the quality of nectar in natural honeybee behavior).

$$fitness_i = \frac{1}{W_i(\{A_j\})} \tag{13}$$

where

$fitness_i$ = *fitness* of the i-th solution

Then, the probability of the employed bees being selected by onlooker bees is calculated. In the ABC algorithm, the selection probability is representative of the waggle dance of natural bees.

$$P_i = \frac{fitness_i}{\sum_{i=1}^{NP} fitness_i} \tag{14}$$

where

P_i is the selection probability for the i-th solution

After calculating the probability of selection, the solution that has a greater possibility based on the probability is chosen by the onlooker bees. Then, the selected solution and its neighborhood are investigated, and the best solution based on the level of fitness is selected (this step is proportional

to the onlooker bees). The onlooker bees choose the desired flowers based on the selection probability that is obtained from the employed bees' dancing, which results in selecting a flower with the best quality of nectar through a search in the neighborhood of the desired flowers. The best solution is stored among available solutions in the feasible space. Knowing that some solutions may not show any improvement (this means that each solution that is compared with generated solutions in the neighborhood has a better fitness level during a specified number of iterations (LIMIT)), these solutions would be replaced with a new feasible solution by Eq. (10) (this represents the behavior of scout bees).

5. Pseudo-code for the ABC algorithm

In this study, a combination of Sonmez's pseudo-code for the ABC algorithm (Sonmez 2011a, b) and Karaboga's constrained algorithm (Karaboga and Akay 2011) is used, and the fly-back mechanism technique is applied instead of Deb's rule for constraint-handling. So selection of solutions from initial feasible population in fly-back mechanism comparing to selection of solutions from initial population with feasible and infeasible solutions that is used in Deb's rule increases the rate of convergence. The steps of the method are as follows.

1. Determine the total number of bees (population size) *NP*, the maximum number of cycles (MNC), and the LIMIT and modification rate control parameter (MR).
2. An initial population is randomly generated using the Eq. (7).
3. Use the Eq. (10) to fix all of the variables from the previous stage to the nearest permissible discrete value in the profile list.
4. Analyze the structures and obtain the response of structures under loads.
5. Check whether any of the existing solutions in the population (food source) satisfy the constraints; otherwise, new solutions must be produced while the constraints are satisfied.
6. Solve the Eqs. (13) and (1) to find the fitness and weight of the structure, respectively, for each of the solutions.
7. Consider half of the number of weights, which are lightest, as employed bees. $SN = NP/2$
8. Select the best (lightest) weight among all existing weights.
9. Set $cycle = 1$
10. Repeat the following loop for each of the employed bees $i = 1, 2, 3, \ldots, SN$ (Steps 11-15).
11. If *MR* is greater than a random number between 0 and 1 for each variable $j = 1, 2, 3, \ldots, D$, the variable is replaced with a new variable in the neighborhood using Eq. (11).
12. If no variables in the 11th step are changed, one of them will be randomly chosen and replaced with the value of the neighborhood by Eq. (12).
13. All of the variables that are obtained in Steps 11 or 12 are rounded to the nearest allowed value in the list of available profiles.
14. Solve the Eqs. (13) and (1) to find the fitness and the structure's weight, respectively.
15. If the fitness level of the solution is better than the previous one, analyze the structure and obtain the response of the structure under loads. Then, if the new solution satisfies the constraints, it will replace the previous solution.
16. The probability of each of the employed bees $i = 1, 2, 3, \ldots, SN$, which is chosen by onlooker bees, is calculated using Eq. (14).
17. Repeat the following loop for each of the onlooker bees $i = 1, 2, 3, \ldots, SN$ (Steps 18-23).

18. If the probability of choosing the food source (the solution) identified by the first employed bee was more than a random value between 0 and 1, an onlooker bee would select the food source. Otherwise, the onlooker bee surveys the probability of choosing the food source by the other employed bees until the constraint has been satisfied.
19. If *MR* is greater than a random number between 0 and 1 for each variable $j = 1, 2, 3, \ldots, D$, the variable is replaced with a new variable in the neighborhood using Eq. (11).
20. If no variables in the 19th step are changed, one of them will be randomly chosen and replaced with the value of the neighborhood by Eq. (12).
21. All variables that are obtained in Steps 19 or 20 are rounded to the nearest allowed value in the list of available profiles.
22. Solve the Eqs. (13) and (1) to find the fitness and the structure's weight, respectively.
23. If the fitness level of the solution is better than the previous one, analyze the structure and obtain the response of the structure under loads. Then, if the new solution satisfies the constraints, it will replace the previous solution.
24. Select and update the best (lightest) weight among all existing weights.
25. Replace a randomly selected solution among the solutions, that shows no improvement within a certain number of iterations (LIMIT) and is not the best solution with a new feasible solution using Eq. (10) (scout bees phase).
26. *cycle* = *cycle* + 1
27. If *cycle* > *MNC*, the process will stop. Otherwise, go to Step 10.

6. Numerical examples

The following three classical test problems were used to check the accuracy of the calculation:

- 10-bar plane truss
- 25-bar space truss
- 72-bar space truss

In all cases, the structure is analyzed using the displacement method. The optimization algorithm and the structural analysis were programmed using MATLAB.

In this study, the ABC algorithm parameters in Examples 1 and 2 were set as follows: size of the bee colony $NP = 50$, $MNC = 516$, and $LIMIT = MNC/3$. In Example 3, the parameters were set as follows: $NP = 50$, $MCN = 1000$, and $LIMIT = MNC/3$. To achieve the best, worst, and average solutions for each problem, twenty independent runs were performed.

6.1 Ten-bar plane truss

The 10-bar plane truss in Fig. 1 is a typical problem in the optimal design of structures, and the efficiency of different optimization methods can be evaluated with this method. The configuration and element numbers of the structure are given in Fig. 1. A single loading condition $p_1 = 100$ kips (445.374 kN) was applied. All of the members' material properties are as follows: an elastic modulus of $E = 10{,}000$ ksi (68.971 GPa) and a mass density of $\rho = 0.10$ lb/in^3 (2,678 kg/m^3). In this problem, there are10 design variables that are selected from a set of 42 discrete values:

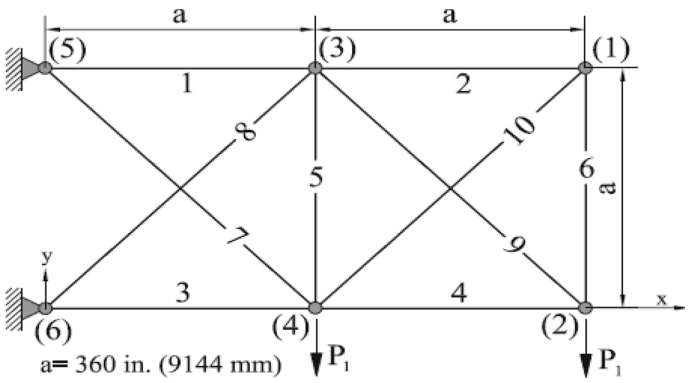

Fig. 1 Ten-bar plane truss

L = {1.62, 1.8, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.35, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.8, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5} in^2.

The free nodes' displacements in both directions must be less than ±2 in (±50.8 mm), and the allowable stress was set to ±25 ksi (±172.25 MPa).

Table 1 shows that the ABC algorithm gives the best solution in eighteen out of 20 design runs. The average weight of the ABC algorithm was 5491.9 (lb) with a standard deviation of 5.047 (lb). The difference between the best and worst solutions developed by this algorithm was 0.41%. The results are better than those obtained using Sonmez's ABC algorithm (Sonmez 2011a). It is assumed that MR = 0.7 in each of the 20 runs.

Fig. 2 is a weight-iteration graph that represents the high speed of convergence of this algorithm; the optimal solution was found in 59 iterations in the best case. The average of the number of truss analyses in the twenty runs to achieve the optimal solution was 11,050 with an average number of iterations of approximately 221. In comparison, the average number of analyses was 25,800 for ABC (Sonmez 2011a), 10,000 for ACO (Camp and Bichon 2004), and 50,000 for HPSO (Li *et al.* 2009). The number of analyses required by this study is larger than that required by ACO (Camp and Bichon 2004), which produced the same best design result. In Table 2, the effect of the modification rate control parameter on the results for the optimum design is shown. As observed, the standard deviations (SD) of 10 runs and the dispersion solutions decreased with an increase in

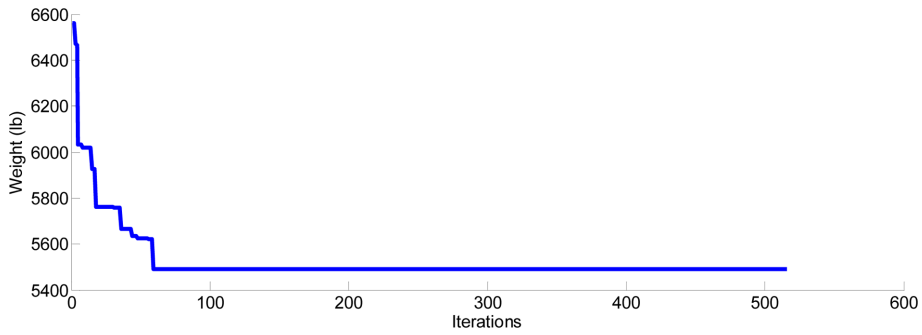

Fig. 2 Weight-Iteration graph for the ten-bar plane truss 1 lb = 0.45359 kg

Table 1 Results of the optimal design of the 10-bar plane truss

Optimal cross-sectional area (in^2)						Element group	
This study	ABC (Sonmez 2011a)	ACO (Camp and Bichon 2004)	HPSO (Li *et al.* 2009)	SA (Kripka 2004)	GA (Rajeev and Krishnamoorthy 1992)		
33.50	33.50	33.50	30.00	33.50	33.50	A_1	1
1.62	1.62	1.62	1.62	1.62	1.62	A_2	2
22.90	22.90	22.90	22.90	22.90	22.00	A_3	3
14.20	14.20	14.20	13.50	14.20	15.50	A_4	4
1.62	1.62	1.62	1.62	1.62	1.62	A_5	5
1.62	1.62	1.62	1.62	1.62	1.62	A_6	6
7.97	7.97	7.97	7.97	7.97	14.20	A_7	7
22.90	22.90	22.90	26.50	22.90	19.90	A_8	8
22.90	22.90	22.90	22.00	22.90	19.90	A_9	9
1.62	1.62	1.62	1.62	1.62	2.62	A_{10}	10
5,490.74	5,490.74	5,490.74	5,531.98	5,490.74	5,613.84	Best (lb)	
5,491.90	5,510.35	N/A	N/A	N/A	N/A	Average (lb)	
5,513.32	5,536.97	N/A	N/A	N/A	N/A	Worst (lb)	
11,050	25,800	10,000	50,000	N/A	N/A	Evaluation (#)	
None	None	None	None	None	None	Constraint violation	

1 $in.^2 = 6.452\ cm^2$ and
1 lb = 0.45359 kg

Table 2 Results of the optimal design as a function of the modification rate control parameter (MR) for the ten-bar plane truss

MR	Best solution (lb)	Worst solution (lb)	Average solution of 10 runs (lb)	Standard deviation (SD) (lb)	Number of runs that found an optimal solution
0	5594.90	6018.80	5758.60	107.156	0
0.1	5502.50	5694.40	5579.70	52.025	0
0.2	5491.70	5568.40	5527.51	29.268	0
0.3	5490.74	5540.40	5506.29	20.558	3
0.4	5490.74	5527.30	5496.17	12.174	7
0.5	5490.74	5507.80	5492.51	5.382	8
0.6	5490.74	5490.74	5490.74	0	10
0.7	5490.74	5490.74	5490.74	0	10
0.8	5490.74	5513.32	5492.96	7.147	9
0.9	5490.74	5491.72	5490.80	0.316	9
1.0	5490.74	5534.70	5495.10	13.914	9

1 lb = 0.45359 kg

the *MR* value from 0 to 0.7. In addition, the optimal solution was found at a higher number of iterations. The changes were not uniform from 0.7 to 1. In this example, considering the results in Table 2, *MR* = 0.7 was chosen.

6.2 The 25-bar space truss

Fig. 3 shows the configuration and element numbers of the 25-bar space truss. All of the members' material properties were an elastic modulus of $E = 10{,}000$ ksi (68.971 GPa) and a mass density of $\rho = 0.10$ lb/in^3 (2,768 kg/m^3). A single load was applied to the structure as shown in Table 3. The members were subjected to stress limitations of ±40 ksi (275.6 MPa). In this problem, there were 8 design groups of variables that were selected from a set of discrete values as follows

$$L = \{0.1, 0.2, 0.3, \ldots, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4\} \ (\text{in}^2)$$

The all-direction displacements at the nodes 1 and 2 must be less than ±0.35 in (±48.89 mm).

Table 4 shows the element's groups and the best, average and worst results; it also compares these results with other algorithms.

As the results show, the difference between the best and the worst results was zero. The average

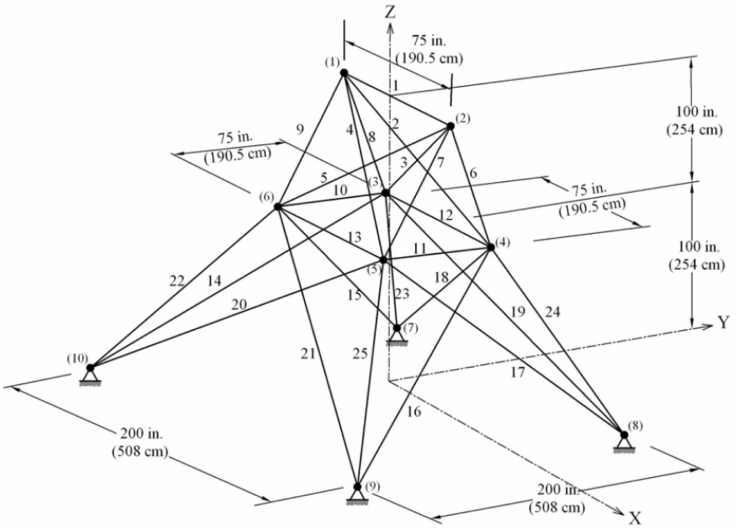

Fig. 3 The 25-bar space truss

Table 3 Nodal loading components (kips) for the 25-bar space truss

Node	Directions		
	x	*y*	*z*
1	1.0	-10.0	-10.0
2	0	-10.0	-10.0
3	0.5	0	0
6	0.6	0	0

1 kips = 4.45 kN

Table 4 Results of the optimal design of the 25-bar space truss

Optimal cross-sectional area (in^2)						Element group	
This study	ABC (Sonmez 2011a)	ACO (Camp and Bichon 2004)	HPSO (Li *et al.* 2009)	SA (Kripka 2004)	GA Rajeev and Krishnamoorthy 1992)		
0.1	0.1	0.1	0.1	0.1	0.1	A_1	1
0.3	0.3	0.3	0.3	0.4	1.8	$A_2 \sim A_5$	2
3.4	3.4	3.4	3.4	3.4	2.3	$A_6 \sim A_9$	3
0.1	0.1	0.1	0.1	0.1	0.2	$A_{10} \sim A_{11}$	4
2.1	2.1	2.1	2.1	2.2	0.1	$A_{12} \sim A_{13}$	5
1.0	1.0	1.0	1.0	1.0	0.8	$A_{14} \sim A_{17}$	6
0.5	0.5	0.5	0.5	0.4	1.8	$A_{18} \sim A_{21}$	7
3.4	3.4	3.4	3.4	3.4	3.0	$A_{22} \sim A_{25}$	8
484.85	484.85	484.85	484.85	484.330	546.010	Best (lb)	
484.85	484.94	486.46	N/A	N/A	N/A	Average (lb)	
484.85	485.05	N/A	N/A	N/A	N/A	Worst (lb)	
10,200	24,250	7,700	25,000	40,000	840	Evaluation (#)	
None	None	None	None	193.8×10^{-6}	None	Constraint violation	

1 $in.^2 = 6.452\ cm^2$ and
1 lb = 0.45359 kg

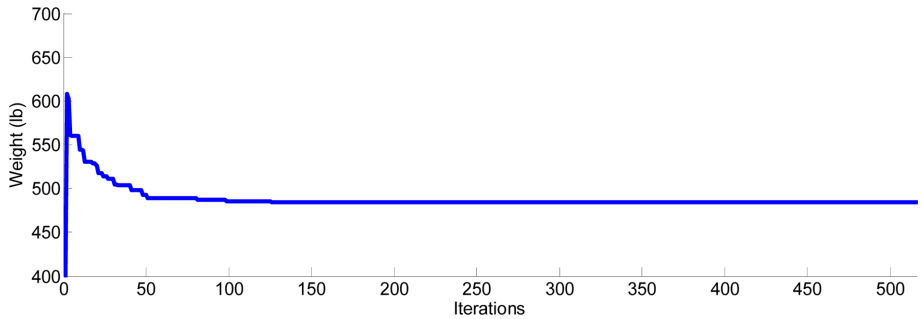

Fig. 4 Weight-Iteration graph for the 25-bar space truss 1 lb = 0.45359 kg

weight determined by the ABC algorithm was 484.85 (lb) with a standard deviation of zero. Fig. 4 shows that the optimal solution was found in 125 iterations in the best case. The average number of truss analyses in twenty runs to achieve the optimal solution was 10,200 with an average number of iterations of approximately 204. For comparison, the average number of truss analyses was 24,250 for ABC (Sonmez 2011a), 40,000 for SA (Kripka 2004), and 25,000 for HPSO (Li *et al*. 2009). The number of analyses required by this study is larger than ACO (Camp and Bichon 2004) (7700) for the same weight and larger than GA (Rajeev and Krishnamoorthy 1992) (840), but the modified ABC algorithm identified a lighter design. In this example, based on the results of Table 5 and comparing the effect of changing the MR to the number of runs that found an optimal solution, it was assumed that MR = 0.9.The results showed that the optimal solution was 484.85 (lb).

Table 5 Results of the optimal design as a function of the modification rate control parameter (MR) for the 25-bar space truss

MR	Best solution (lb)	Worst solution (lb)	Average solution of 10 runs (lb)	Standard deviation (SD) (lb)	Number of runs that found an optimal solution
0	499.9492	517.2931	510.3282	6.247	0
0.1	498.1343	522.2967	505.9005	7.862	0
0.2	485.9052	499.2220	490.0912	4.027	0
0.3	485.3797	487.3456	486.3550	0.688	0
0.4	485.0488	488.5146	485.9462	1.140	0
0.5	484.8542	485.3797	485.0624	0.127	1
0.6	484.8542	485.0488	484.8931	0.082	8
0.7	484.8542	485.0488	484.8931	0.082	8
0.8	484.8542	485.0488	484.8736	0.061	9
0.9	484.8542	484.8542	484.8542	0	10
1.0	484.8542	485.0488	484.8736	0.061	9

1 lb = 0.45359 kg

Compared with other algorithms, only the SA algorithm (Kripka 2004) was able to identify a lighter-weight solution. By changing the displacement constraint from 0.35 (in) to 0.3501 (in), the ABC algorithm was able reduce the weight of the truss to 484.33 (lb), which is exactly equal to the result of the SA algorithm.

6.3 The 72-bar space truss

The 72-bar skeletal tower is shown in Fig. 5. The truss was subjected to two distinct loading conditions as shown in Table 6. The displacements at the uppermost portion of 1, 2, 3, and 4 in the x-direction and y-direction were less than ±0.25 in (±6.35 mm).

The allowable stresses for all of the members were ±25 ksi (±159.125 MPa). The members' material properties were an elastic modulus of $E = 10{,}000$ ksi (68.971 GPa) and a mass density of $\rho = 0.10$ lb/in^3 (2,768 kg/m^3).

In this problem, there were 16 design groups of variables that were selected from a set of 2,901 discrete values from 0.1 to 3.00 in^2 with a 0.0010 in^2.

Table 7 shows the element groups and the best, average, and the worst results and compares these results with the other algorithms. The difference between the best and the worst answers was 0.0268% with a standard deviation of 0.028 (lb). In this example, as shown in Fig. 6, the ABC algorithm found an optimal solution of 369.669 (lb) after 746 iterations in the best case. The average number of truss analyses in the twenty runs that were required to achieve the optimal solution was 37,950 with approximately 759 iterations. In comparison, the average number of truss analyses was 50,000 for ABC (Sonmez 2011a) and 50,000 for HPSO (Li *et al*. 2009). It is noteworthy that the proposed algorithm required an average of 8100 analyses to achieve a value close to optimal solution from Sonmez's ABC algorithm. The number of analyses required by this study was larger than the number required by ACO (Camp and Bichon 2004) (18,500) and GA

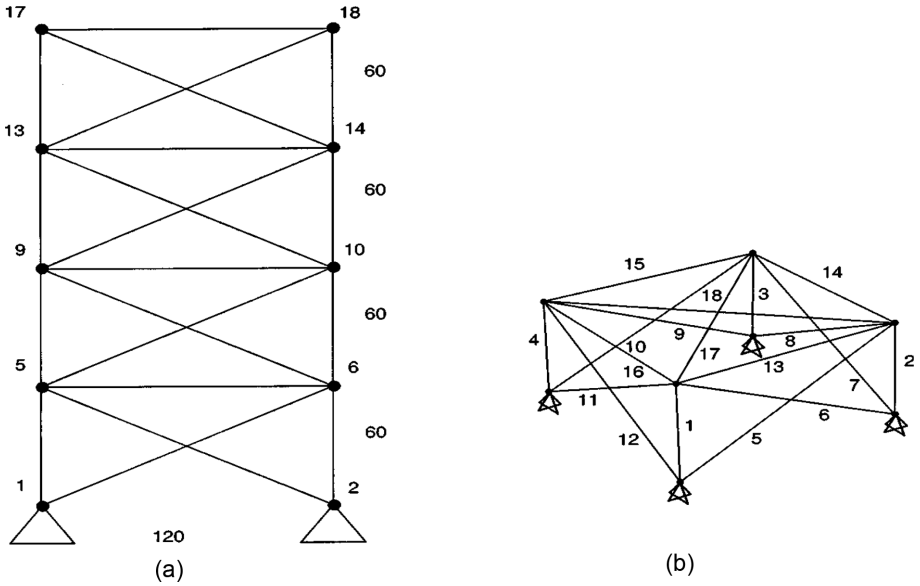

Fig. 5 The 72-bar space truss: (a) dimensions and node numbering scheme and (b) element numbering pattern for the first story (Camp and Bichon 2004) (1 in. = 2.54 cm)

Table 6 Nodal loading conditions (kips) for the 72-bar space truss

Loading conditions	Node	*x*	*y*	*z*
1	1	5	5	-5
2	1	0	0	-5
2	2	0	0	-5
2	3	0	0	-5
2	4	0	0	-5

1 lb = 0.45359 kg

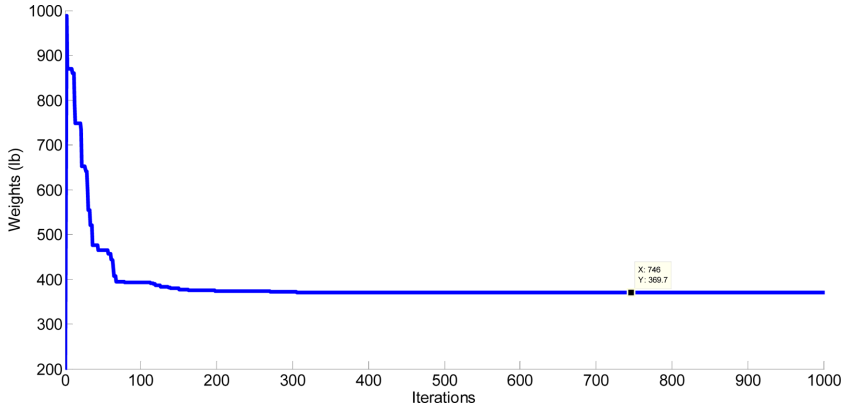

Fig. 6 Weight-Iteration graph for the 72-bar space truss 1 lb = 0.45359 kg

Table 7 Results of the optimal design of a 72-bar space truss

Optimal cross-sectional area (in^2)					Element group	
This study	ABC (Sonmez 2011a)	HPSO (Li *et al*. 2009)	ACO (Camp and Bichon 2004)	GA (Erbatur *et al.* 2000)		
1.845	0.156	2.100	1.980	0.155	$A_1 \sim A_4$	1
0.507	0.553	0.600	0.508	0.535	$A_5 \sim A_{12}$	2
0.100	0.391	0.100	0.101	0.480	$A_{13} \sim A_{16}$	3
0.100	0.597	0.100	0.102	0.520	$A_{17} \sim A_{18}$	4
1.261	0.520	1.400	1.303	0.460	$A_{19} \sim A_{22}$	5
0.509	0.515	0.500	0.511	0.530	$A_{23} \sim A_{30}$	6
0.100	0.101	0.100	0.101	0.120	$A_{31} \sim A_{34}$	7
0.100	0.103	0.100	0.100	0.165	$A_{35} \sim A_{36}$	8
0.489	1.271	0.500	0.561	1.155	$A_{37} \sim A_{40}$	9
0.508	0.512	0.500	0.492	0.585	$A_{41} \sim A_{48}$	10
0.100	0.100	0.100	0.100	0.100	$A_{49} \sim A_{52}$	11
0.100	0.100	0.100	0.107	0.100	$A_{53} \sim A_{54}$	12
0.100	1.843	0.200	0.156	1.755	$A_{55} \sim A_{58}$	13
0.520	0.517	0.500	0.550	0.505	$A_{59} \sim A_{66}$	14
0.393	0.102	0.300	0.390	0.105	$A_{67} \sim A_{70}$	15
0.535	0.100	0.700	0.592	0.155	$A_{71} \sim A_{72}$	16
369.669	379.893	388.94	380.24	385.760	Best (lb)	
369.726	380.053	N/A	383.16	N/A	Average (lb)	
369.769	380.173	N/A	N/A	N/A	Worst (lb)	
37,950	50,000	50,000	18,500	840	Evaluation (#)	
None	None	None	193.8×10^{-6}	None	Constraint violation	

1 $in.^2 = 6.452\ cm^2$ and
1 lb = 0.45359 kg

(Erbatur *et al*. 2000) (840), but the resulting solution was lighter than the solutions produced by these algorithms. Compared with the other results, the obtained result was the best and the lightest design.

Table 8 shows the effect of changing the MR to the number of runs that determined the optimal solution. Because the list of allowed variables covered a wide range of variables and the distance between them was very small, in each run of the algorithm, the optimal weight may differ only by a decimal point. Therefore, we considered numbers with a decimal point less than 0.8 as the optimal solution. In this example, according to Table 8, an $MR = 0.7$ was assumed.

7. Conclusions

In this paper, the discrete optimization of trusses using a previously developed ABC algorithm

Table 8 Results of the optimal design as a function of the modification rate control parameter (MR) for the 72-bar space truss

MR	Best solution (lb)	Worst solution (lb)	Average solution of 10 runs (lb)	Standard deviation (SD) (lb)	Number of runs that found an optimal solution
0	395.4259	489.1245	442.7402	28.5432	0
0.1	372.1467	374.4479	373.3677	0.7063	0
0.2	369.8825	370.6810	370.1346	0.2334	0
0.3	369.7306	369.9147	369.8145	0.0582	6
0.4	369.6770	369.8730	369.7566	0.0589	8
0.5	369.6932	369.8103	369.7353	0.0372	9
0.6	369.6852	369.8071	369.7297	0.0379	9
0.7	369.6877	369.7426	369.7145	0.0194	10
0.8	369.6733	369.7817	369.7319	0.0337	10
0.9	369.7115	379.8166	369.7369	0.3036	9
1.0	369.6946	369.7863	369.7516	0.0303	10

1 lb = 0.45359 kg

and its effectiveness at modeling three standard examples, a plane truss and two space trusses was studied and suggested. The algorithm was programmed using MATLAB software and was modified as follows:

- Using the ABC algorithm presented by Karaboga and Akay (2011) to solve numerical problems.
- Using Sonmez's proposed process (Sonmez 2011a) to select half of the best results in the initial population as employed bees.
- Converting the continuous ABC algorithm to a discrete algorithm inspired by Kaveh's and Talatahari's proposed process (Kaveh and Talatahari 2007).
- Appling constraint-handling to the problem. Thus far, penalty functions and Deb's rule have been used to apply constraint-handling in the ABC algorithm. However, this paper used the fly-back mechanism technique to apply constraint- handling.

A brief summary of the results of this study is as follows:

- Obtained a result from the desired algorithm and compared it with the other optimization algorithms. In some cases, it produced the same results, and in other applications, it was significantly better.
- The difference between the best and the worst result was too small, and in some cases, the difference equaled zero within twenty runs, so the proposed algorithm achieved the optimum result in all twenty runs.
- Using the constraint-handling fly-back mechanism technique significantly increased the convergence speed and the accuracy of the optimum result compared with the other techniques applied in the ABC algorithm, such as penalty functions and Deb's rule.

The ability to reduce the structural weight and the design time proves that this algorithm is one of the most powerful algorithms available for structural truss weight optimization.

References

Akay, B. and Karaboga, D. (2012), "A modified artificial bee colony algorithm for real- parameter optimization", *Inform. Sci.*, **192**, 120-142.

Brajevic, I., Tuba, M. and Subotic, M. (2011), "Performance of the improved artificial bee colony algorithm on standard engineering constrained problems", *Int. J. Math. Comput. Simul*., **5**(2), 135-143.

Cagatay, I.H., Dundar, C. and Aksogan, O. (2003), "Optimum design of prestressed concrete beams by a modified grid search method", *Struct. Eng. Mech.*, **15**(1), 39-52.

Camp, C.V. and Bichon, B.J. (2004), "Design of space trusses using ant colony optimization", *J. Struct. Eng., ASCE*, **130**, 741-751.

Chen, S.Y. and Rajan, S.D. (2000), "A robust genetic algorithm for structural optimization", *Struct. Eng. Mech.*, **10**(4), 313-336.

Consoli, S. (2006), *Combinatorial Optimization and Metaheuristics*, Operational Research Report, School of Information Systems, Computing and Mathematics Brunel University.

Deb, K. (2000), "An efficient constraint handling method for genetic algorithms", *Comput. Meth. Appl. Mech. Eng.*, **186**(2-4), 311-338.

Erbatur, F., Hasancebi, O., Tutuncu, I. and Kılıc, H. (2000), "Optimum design of planner and space structures with genetic algorithm", *Comput. Struct.*, **75**, 209-224.

Geem, Z.W. (2010), *Harmony Search Algorithms for Structural Design Optimization*, Springer.

Hadidi, A., Kazemzadeh Azad, S. and Kazemzadeh Azad, S. (2010), "Structural optimization using artificial bee colony algorithm", *Proceedings of the 2nd International Conference on Engineering Optimization*, Lisbon, Portugal.

Hooke, R. and Jeeves, A. (1961), "Direct search solution of numerical and statistical problems", *J. ACM*, **8**, 212-229.

Kang, F., Li, J., Li, H., Ma, Z. and Xu, Q. (2010), "An Improved Artificial Bee Colony Algorithm", *Intelligent Systems and Applications (ISA) 2nd International Workshop on:* 1-4, Wuhan.

Karaboga, D. (2005), *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report, TR06, Erciyes University.

Karaboga, D. and Akay, B. (2009), "A comparative study of artificial bee colony algorithm", *Appl. Math. Comput.*, **214**, 108-132.

Karaboga, D. and Basturk, B. (2007), "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", *J. Global Opt.,* **39**(3), 459-471.

Karaboga, D. and Akay, B. (2011), "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems", *Appl. Soft Comput.*, **11**, 3021-3031.

Kaveh, A. and Talatahari, S. (2007), "A discrete particle swarm ant colony optimization for design of steel frames", *Asian J. Civil Eng. (Build. Hous.)*, **9**(6), 563-575.

Kaveh, A. and Talatahari, S. (2008), "A hybrid particle swarm and ant colony optimization for design of truss structures", *Asian J. Civil Eng. (Build. Hous.)*, **9**(4), 329-348.

Kaveh, A. and Talatahari, S. (2009), "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures", *Comput. Struct.,* **87**, 267-283.

Kim, S.E., Song, W.K. and Ma, S.S. (2004), "Optimal design using genetic algorithm with nonlinear elastic analysis", *Struct. Eng. Mech.*, **17**(5), 707-725.

Kripka, M. (2004), "Discrete optimization of trusses by simulated annealing", *J. Braz Soc Mech. Sci. Eng.*, **26**(2), 170-173.

Lee, E.H. and Park, J. (2011), "Structural design using topology and shape optimization", *Struct. Eng. Mech.*, **38**(4), 517-527.

Lemmens, N., Jong, S. and Tuyls, K. (2007), "A bee algorithm for multi-agent systems: recruitment and navigation combined", *Proceeding of ALAG 2007, An AAMAS'07 Workshop*, Honolulu, Hawaii.

Li, L.J., Huang, Z.B. and Liu, F. (2009), " A heuristic particle swarm optimization method for truss structures with discrete variables", *Comput. Struct.*, **87**, 435-444.

Rajeev, S. and Krishnamoorthy, C.S. (1992), "Discrete optimization of structures using genetic algorithms", *J. Struct. Eng., ASCE*, **118**(5), 1233-1251.

Luo, P., Pan, T.S., Tsai, P. and Pan, J.S. (2010), "Parallelized artificial bee colony with ripple-communication strategy", *ICGEC2010*, 350-353.
Sonmez, M. (2011), "Discrete optimum design of truss structures using artificial bee colony algorithm", *Struct. Multidiscip. O.*, **43**(1), 85-97.
Sonmez, M. (2011), "Artificial bee colony algorithm for optimization of truss structures", *Appl. Soft Comput.*, **11**(2), 2406-2418.
Togan, V., Daloglu, A.T. and Karadeniz, H. (2011), "Optimization of trusses under uncertainties with harmony search." *Struct. Eng. Mech.*, **37**(5), 543-560.
Tsai, P.W., Pan, J.S., Liao, B.Y. and Chu, S.C. (2009), "Enhanced artificial bee colony optimization", *Int. J. Innov. Comput., Inform. Control*, **5**(12), 5081-5092.
Yang, X.S. (2008), *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, United Kingdom.
Yun, Y.M., Kang, M.M. and Lee, M.S. (2006), "Optimum design of plane steel frames with PR-connections using refined plastic hinge analysis and genetic algorithm", *Struct. Eng. Mech.*, **23**(4), 387-407.
Zhu, G. and Kwong, S. (2010), "Gbest-guided artificial bee colony algorithm for numerical function optimization", *Appl. Math. Comput.*, **217**, 3166-3173.