Applied Soft Computing xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

Artificial Bee Colony algorithm for optimization of truss structures

Mustafa Sonmez*

Department of Civil Engineering, School of Engineering, Aksaray University, Aksaray 68100, Turkey

ARTICLE INFO

Article history:
Received 6 April 2009
Received in revised form 21 July 2010
Accepted 28 September 2010
Available online xxx

Keywords:
Artificial Bee Colony
Truss optimization
Structural design
Adaptive penalty function
Size optimization

ABSTRACT

The main goal of the structural optimization is to minimize the weight of structures while satisfying all design requirements imposed by design codes. In this paper, the Artificial Bee Colony algorithm with an adaptive penalty function approach (ABC-AP) is proposed to minimize the weight of truss structures. The ABC algorithm is swarm intelligence based optimization technique inspired by the intelligent foraging behavior of honeybees. Five truss examples with fixed-geometry and up to 200 elements were studied to verify that the ABC algorithm is an effective optimization algorithm in the creation of an optimal design for truss structures. The results of the ABC-AP compared with results of other optimization algorithms from the literature show that this algorithm is a powerful search and optimization technique for structural design.

1. Introduction

Over the last 60 years, a number of optimization techniques have been developed and used in the structural optimization [1,2]. These techniques can be broadly divided into two groups: (i) gradient based and (ii) direct search (stochastic or non-gradient based). Since there are known difficulties in the application of gradient-based techniques in structural optimization problems, direct search techniques have gained popularity in recent years [2–7]. Direct search techniques explore the design space by generating a number of successive solutions to guide the algorithm to an optimal design. Genetic algorithms [8–12], simulated annealing algorithms [13–16] evolutionary programming [17] and evolutionary strategies [18] are the most notable direct search optimization techniques used in the solution of engineering problems. The main characteristic of these algorithms is the imitation of biological and physical events by evolving a good enough or near-optimal solution over a number of successive iterations. These techniques do not require the evaluation of gradients of objective and constraint functions, but they do require a significant amount of computer power. In the past, such techniques were considered impractical for design use due to the limitations of earlier computers. Recent advancements in computer hardware, especially in memory size and the speed of personnel computers make direct search techniques more feasible and practical.

More recently, another branch of biologically inspired algorithms have attracted the attention of researchers all over the world. Algorithms belonging to this field imitate the collective behavior of a group of social insects (for example, bees, termites, ants and wasps) to solve complex optimization problems. These social insects live closely together in their nest and divide up the various tasks within the colony, such as foraging, nest building and defense. Each member of the colony performs their own tasks by interacting or communicating directly or indirectly in their local environment. Even when one or more individuals fail to carry out their task, the group as whole can still perform their tasks [19,20]. The process of the division of labor among insects is believed to be more effective than individual action performed by an individual insect. These collective and adaptive behaviors of simple insects have been used by researchers to develop new optimization algorithms, known as swarm-based optimization algorithms. Particle swarm optimization [21–24] and ant colony optimization algorithms [25,26] are well known swarm-based algorithms and are already employed to solve structural optimization problems. On the hand, bee-inspired algorithms have not yet been employed to solve structural engineering optimization problems. The main objective of this paper is to propose a bee-based algorithm for the optimum design of planer and space trusses consisting of continuous design variables and to evaluate the performance of the algorithm by comparing the results of the algorithm with those of other optimization techniques. Also, modifications that have been made to implement the algorithm to the structural optimization are described.

The bee-inspired optimization algorithms are based on either a crude imitation of their mating process or their foraging behaviors. The algorithms based on the biological process of their

* Tel.: +90 382 280 1347; fax: +90 382 280 1365.
E-mail addresses: mustafasonmez@aksaray.edu.tr, mustafasonmez58@gmail.com.

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.09.003

reproduction are generally used in the combinatorial optimization problems while algorithms based on the foraging behavior of honeybees are used for various types of optimization problems. The Bee Colony Optimization [27,28], Virtual Bee [29], Bee [30,31] and Artificial Bee Colony [32–35] are some of the algorithms based on mimicking the foraging behavior of honeybee swarms. Although all bee algorithms share some common features, they do have some different characteristics.

The Bee Colony Optimization (BCO) algorithm developed by Teodorovic and co-workers [27,28] was used to solve the traveling salesman problem and a number of other numerical examples. In addition, they presented some potential application areas of the BCO algorithm in transportation and traffic engineering problems. Teodorovic [27] also stated that the BCO algorithm based on Swarm Intelligence principles gave encouraging results for its use in solving complex engineering problems. Yang [29] proposed the Virtual Bee Algorithm (VBA) and demonstrated how it could solve two-dimensional numerical problems. Based on his findings, Yang stated that VBA was usually as effective as genetic algorithms and could, in some cases, optimize more effectively than a conventional algorithm due to the parallelism of the multiple agents.

The Bee Algorithm (BA) originally proposed by Pham et al. [30,31] is used for solving unconstrained function optimization problems and training multi-layered perceptron networks to recognize different patterns in control charts. They claimed that the BA generally gives better results than the genetic algorithm and the ant colony algorithm, when compared with the BA in terms of speed of optimization and accuracy of the results. However, one of the drawbacks of the BA is the number of parameters that must be tuned before executing the algorithm.

Karaboga and Basturk [32–34] proposed the Artificial Bee Colony (ABC) algorithm. They used the ABC algorithm to solve unconstrained and constrained function optimization problems. The performance of the ABC algorithm was compared to that of differential evaluation, particle swarm optimization and an evolutionary algorithm. Karaboga and Basturk declared that the ABC algorithm when compared with differential evaluation, particle swarm optimization and an evolutionary algorithm performed better and could be effectively employed to solve engineering problems. Recently, Singh [35] used the ABC algorithm to solve the leaf-constrained minimum spanning tree discrete optimization problems. He compared the ABC algorithm with three other metaheuristic algorithms, namely the genetic algorithm, ant colony optimization algorithm and tabu search algorithm. Singh stated that the results of the new ABC algorithm outperforms all the other approaches and provides quality solutions in shorter time.

Since the ABC algorithm has been shown to perform well, it was selected for use in the present study for truss optimization with some deviations. Similar to other direct search algorithms, the ABC is an unconstrained optimization algorithm. To accommodate the inclusion of constraints, Karaboga and Basturk [33] proposed the Deb's selection mechanism. In this work, the self-adaptive penalty function approach is used to find a way of incorporating constraints in order to improve the ABC algorithm. Although only the size optimization of truss structures is considered in this study, it is believed that the ABC optimization algorithm can also be used for the topology and shape optimization of other types of structures.

The remainder of this paper is arranged as follows: Section 2 briefly presents the characteristics of the structural optimization problems. In Section 3, the natural forging behavior of real bees is described, Section 4 describes the modeling of foraging behavior of artificial bees, the constraint handling procedure included in the ABC algorithm is given in Section 5. The pseudo-code of ABC-AP algorithm is presented in detail in Section 6. The analysis of standard test problems to demonstrate the effectiveness of the algorithm in finding the optimal solution is given in Section 7. Finally, Section 8 presents the conclusions.

2. The presentation of the structural optimization problem

Many problems in engineering have multiple solutions and selecting the appropriate one can be a major task. In sizing optimization problems, the major task is to find an optimal cross-section of the elements of a system by minimizing the weight of the structure. This is expressed mathematically as:

$$\text{Minimize } W(\vec{A}) = \sum_{j=1}^{n} A_j L_j \rho_j \tag{1}$$

where A_j, L_j, and ρ_j are the cross-sectional area, length and unit weight of jth truss member, respectively; $W(\vec{A})$ is the weight of truss which is minimized; n is the total number of members. The vector $\vec{A}$ represents element cross-section vector that is selected between lower A^l and upper A^u bounds. Eq. (1) is subjected to the following normalized constraints:

$$s_{m,l}(\vec{A}) = \frac{\sigma_{m,l}}{\sigma_{m,allowed}} - 1 \le 0, \quad m = 1, 2, \ldots, n \tag{2}$$

$$b_{m,l}(\vec{A}) = \frac{\lambda_{m,l}}{\lambda_{m,allowed}} - 1 \le 0, \quad m = 1, 2, \ldots, n \tag{3}$$

$$d_{k,l}(\vec{A}) = \frac{u_{k,l}}{u_{k,allowed}} - 1 \le 0, \quad k = 1, 2, \ldots, n_n \tag{4}$$

where $s_{m,l}$, $b_{m,l}$, and $d_{k,l}$ are respectively, the element stress, element buckling and nodal displacement constraint functions; $\sigma_{m,l}$ and $\lambda_{m,l}$ are the stress and the slenderness ratio of mth member due to the loading condition l; $\sigma_{m,allowed}$ and $\lambda_{m,allowed}$ are the allowable axial stress and allowable slenderness ration for mth member, respectively; $u_{k,allowed}$ and $u_{k,l}$ are the allowable displacement and nodal displacement of kth degrees of freedom due to the loading condition l, respectively; n_n is the number of restricted displacements. All the normalized constraint functions are activated when the violated constraints have values larger than zero.

3. The behavior of real honeybees in their natural environment

Honeybees live in social units called colonies, depending on the time of year a typical colony includes a single queen, thousands of semi-sterile female workers and a few thousand males (drones). Adult workers are responsible for executing all the tasks associated with colony living such as; processing and storing food, cleaning cells, feeding larvae (nursing behavior), secreting wax and constructing combs, and guarding the entrance [36]. When the female bees are about 3 weeks old, they begin foraging, cease performing most tasks within the hive and usually remain foragers for the rest of their lives [37]. Foragers are able to utilize a large number of flower nectar (food sources) in multiple directions up to 12 km from the hive, but mostly they fly within a 3 km radius [30].

In a colony, the female bees start the foraging process by randomly searching for the promising flower patches. After finding a food source, the bee loads up with nectar then returns to the hive and unloads her nectar. Then, she may inform her nest mates about her findings through the movements known as the "waggle dance." This dance gives three pieces of information regarding the flower patch; the direction in which it can be found, distance from the hive and quality rating [20,28,31]. In a decentralized but intelligent fashion, some of the bees decide to follow their nest mates who have performed the waggle dance; others, to maximize their nectar intake, search for the food source without following the dancers. This means that each bee can follow one of three options

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

after unloading the nectar: (a) abandon the food source and search for another promising flower patch, (b) continue to forage at the food source without recruiting nest mates, or (c) perform the waggle dance to recruit nest mates before returning to the food source. Each bee follows one of the above options based upon the food level of that nectar source. If a bee finds a nectar source which is above a certain limit, she follows option (c). If the nectar source is average, the bee goes to forage at the food source without recruiting nest mates. Otherwise, the bee continues to search for promising nectar sources as option (a). The main goal of the foragers is to locate the most abundant nectar source [20,28,30].

4. Modeling bee behavior

In the ABC algorithm, each food source exploited by the bees represents a possible solution to given optimization problem. The location and amount of nectar from the flower patch correspond to the design variables and the fitness function, respectively. All the worker bees (N) leave the hive to search for promising flower patches. After the workers bees return to hive with a certain amount of nectar, the first half (SN) that found the best food sources become "employed bees." The remainder of the bees watches the waggle dance to decide which of the employed bees followed. These bees, which watch the waggle dance, are called "unemployed bees" or "onlooker bees."

Each food source has only one employed bee; that is, the number of food sources is equal to the number of employed bees. The number of unemployed bees which will fly to a food source depends on the amount of nectar at the source. The unemployed bees choose a food source according to the quality of the nectar. More unemployed bees prefer to visit an abundant nectar source while fewer or no unemployed bees choose the food source having less nectar than others. It means that unemployed bees select a food source according to a probability proportional to the amount of nectar to be found at the food source [35]. The probability p_i for that source i is calculated in the following way:

$$p_i = \frac{1/{}_iW(\vec{A})}{\sum_{j=1}^{SN} 1/{}_jW(\vec{A})} \tag{5}$$

A candidate food source is created from the neighborhood of the old food source. It means that the ABC algorithm uses the old food source (${}_iA_j^{old}$) to search for a candidate food source (${}_iA_j^{new}$). Numerically, the location of a candidate food source i is determined as:

$$A_j^{new} = {}_iA_j^{old} + \phi({}_iA_j^{old} - {}_kA_j^{old}) \tag{6}$$

where ϕ is a random number between -1 and 1. A_j^{new} is an updated design variable. The left hand subscripts represent the solution number (food source, $i = 1, 2, 3, \ldots, SN$) while the right hand script denote the design variable number ($j = 1, 2, 3, \ldots, D$). k is a randomly chosen integer number but cannot be equal to i. ${}_kA_j^{old}$ plays an important role in the ABC convergence behavior since it is employed to control the exploration abilities of the bees. It directly influences the location of the new food source, which is based on the previous location of other food sources in the regions of the design space. Every employed bee determines a new food source in the neighborhood of its currently associated food source and evaluates the new amount of food as shown in Eq. (1). If the food level in the new location is better than the old one, the new position becomes the food source; otherwise, the old location is maintained as the best food source.

As mentioned above, the ABC algorithm is iterative. If there is no improvement in the amount of nectar from a food source after a predefined iteration (LIMIT), this food source is discarded by its employed bee. These employed bees become "scout bees" acting as the colony's explorers [26]. Concerned primarily with finding any kind of nectar source and they do not have any guidance as to where to look for food. The scouts may accidentally discover rich, entirely unknown food sources and when this happens the scout bee becomes an employed bee. A new location found by a scout bee i is calculated as:

$$A_j^{new} = A_j^l + \lambda(A_j^u - A_j^l) \tag{7}$$

where λ is a random number between 0 and 1. A_j^l and A_j^u are the lower and upper bounds of the jth variable, respectively.

5. Constraint handling procedure

The objective of structural optimization is to develop a design that minimizes the total structural weight while satisfying all design requirements such as member stresses, nodal displacements and member buckling. These requirements correspond to the constraints for the optimization problems. Like other direct search algorithms, the ABC algorithm was originally developed for unconstrained optimization problems, and hence it is necessary to somehow incorporate constrains into the ABC algorithm to solve structural optimization problems. The advantages and disadvantages of the well known constraint handling techniques were presented in a review paper by Coello [38] and Kicinger et al. [17]. In the Coello's research [38], constraint-handling techniques were divided into five major groups (i) penalty functions, (ii) special representations and operations, (iii) repair algorithms, (iv) separation of objectives and constraints, (v) hybrid methods. The traditional approach for handling the design constraints of optimization problems is to use penalty function methods in which a constrained optimization problem is transformed into an unconstrained problem by adding a certain value to the objective function based on the amount of constraint violations.

Most of the penalty function methods require pre-defined or static penalty function coefficients at the beginning of the calculations. These coefficients are generally determined by trial and error methods. If the penalty function coefficients are too low, the optimized design converges to an unfeasible solution without satisfying the constraints. On the other hand, if the penalty function coefficients are too high, the optimized design will not have a satisfactory objective function value. In order to overcome this drawback of the static penalty function method, various types of penalty functions have been proposed and studied [38,17]. Adaptive penalty is one of the methods, in which a penalty function coefficient is adapted according to feedback received from the search process. Different versions of the adaptive penalty function techniques have been proposed by various researchers including Hadj-Alouane and Bean [39], Nanakorn and Meesomklik [40], Hasancebi [18], and Togan and Daloglu [11].

The original ABC algorithm uses the Deb's selection mechanism [41] that belongs to the category (iv) to accommodate the inclusion of constraints. Deb's method includes three very simple heuristic rules to compare two solutions: (i) a feasible solution is always preferred to an unfeasible one, (ii) between two feasible solutions, the one having a better objective function is preferred, (iii) between two unfeasible solutions, the one having a smaller constraint function is preferred. The advantage of the Deb's selection mechanism is that it does not require any penalty factor because the selection procedure is only performed in a pair wise comparison and is very easy to employ in any optimization problem. On the other hand, this method seems to have problems in maintaining diversity in the population [38]. In addition, an optimal solution generally lies on or close to boundaries between feasible and unfeasible search spaces in the structural optimization when design variables are continu-

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

ous. Deb's method forces the candidate solution to be in the feasible region and a candidate solution may not cross the boundaries. These disadvantages may artificially slow down the ABC algorithm.

Due to the undesirable effect of Deb's method and the static penalty function methods mentioned above, an adaptive penalty function method is considered in this study for constraint handling within the ABC. There are several different versions of the adaptive penalty function methods. The common property of all these methods is that the penalty coefficient is updated for every individual according to information gathered from search space. The adaptive penalty function proposed by Hasancebi [18] is used in this work. The following function is used to transform a constrained optimization problem to an unconstraint optimization problem for ith bee.

$$W_c = {}_iW\left[1 + {}_ir(t)\left[\sum_{j=1}^{l}\left(\sum_{m=1}^{n}(s_{m,j}+b_{m,j}) + \sum_{k=1}^{n_n} d_{k,j}\right)\right]\right] \quad (8)$$

W_c is unconstrained objective function; W is constraint objective function; $s_{m,j}$, $b_{m,j}$ and $d_{k,j}$ are normalized constraints functions activated when their values are larger than zero and $r(t)$ is the penalty coefficient which is used to tune the intensity of penalization and a function of iteration. When $r(t)$ is set to a predefined static value such as $r = 1$ for all cycles, the resulting function becomes a static penalty function. In the adaptive penalty function implementation, the penalty coefficient $r(t)$ is adjusted based on the feedback from the previous solution as in shown Eq. (9)

$$r(t)\begin{cases} 1/f \cdot {}_ir(t-1) & \text{if } {}_iW_c(t-1) \text{ is feasible} \\ f \cdot {}_ir(t-1) & \text{if } {}_iW_c(t-1) \text{ is infeasible} \end{cases} \quad (9)$$

where $r(t)$ and $r(t-1)$ are the penalty coefficients at cycles (t) and $(t-1)$, respectively. $W_c(t-1)$ denotes the last known objective function for ith employed bee and f is the arbitrary constant referred to as the learning parameter of $r(t)$. The value of f depends on the problem to be solved. The best value for f is recommended to be 1.1 for a simulated annealing optimization algorithm [18]. The ABC algorithm is quite different from the simulated annealing algorithm; therefore, the recommended value for f is unsuitable for the ABC algorithm. In this study, f is calculated using the following equation:

$$f = 1 + \frac{1}{n_c} > 1.01 \quad (10)$$

where n_c is the total number of constraints. Eq. (10) has been set empirically. The learning parameter f is designated to adjust the intensity of the penalty parameter r. It is not difficult to imagine that if an optimization problem has a large number of constraints, the total value of the normalized constraints is possible to be a large number. Hence the f should be small. On the contrary if the optimization problem has a small number of constraints, total value of the normalized constraints may be a small number. Hence f is selected to be a function of the total constraints. As mentioned above the optimal solution generally lies on or close to the boundary between the feasible and the unfeasible search space in the structural optimization. Therefore, the optimal solution should be found on the boundary. In the adaptive penalty scheme, if the solution in the preceding iteration is feasible, the penalty coefficient f is reduced to point towards the unfeasible region. On the other hand, if the solution in the preceding iteration is in the unfeasible region, the penalty function is increased to make the feasible region more attractive.

6. The pseudo-code of the proposed ABC-AP algorithm

The algorithm proposed in the previous section can be incorporated into a general truss analysis program to solve optimal design problems. A pseudo-code of the procedure for the application of the algorithm is given as follows:

Step 1. Set the total number of bees (N) and maximum number of cycles (MNC).
Step 2. Generate a random initial bee colony, A_j ($i = 1, 2, \ldots, N, j = 1, 2, 3, \ldots, D$), constructed using Eq. (7).
Step 3. Read input data to construct structure then carry out the structural analysis for the given load condition(s). The joint displacements and member forces obtained from the analysis are used to calculate the values of normalized constraints and the unconstrained weight function (${}_iW$) for every bees.
Step 4. Select the best food locations (SN) among the candidate food sources (N). The bees associated with the best locations become "employed bees".
Step 5. Set cycle = 1;
Step 6. Loop over each food source ($i = 1, 2, \ldots,$ SN)
6.1 Recruit the unemployed bees based on the probability p_i using Eq. (5).
6.2 Produce new solutions for an employed bee and any existing unemployed bee(s), using Eq. (6).
6.3 Carry out the structural analysis under applied load conditions. The joint displacements and member forces are used to calculate the values of normalized constraints and unconstrained weight function (W_i) for each bee.
6.4 Select the best food level for each food source. If the food level in the new location is better than the old one, the new position becomes the food source; otherwise, the old one is maintained as a food source.
Step 7. Discard the food source if the following conditions are true: (i) there is no improvement of the food level after the LIMIT number of cycles, and (ii) the source is not one of the best food sources. Then replace the food source with a random solution using Eq. (7).
Step 8. cycle = cycle + 1.
Step 9. If the cycle is greater than MNC or there is no improvement in the best solution after the number of LIMIT cycles, stop the procedure; otherwise, go to step 6.

The ABC-AP algorithm has some deviations from the original ABC algorithm proposed by Karaboga and Basturk [33]. The original algorithm selects the half of the bees as "employed bees" and generates solutions for each of them. Also, in the original algorithm, every employed bee constructs solutions in their neighborhood and moves these locations if they have better fitness, then unemployed bees construct solutions in the neighborhoods of the employed bees they followed. On the other hand, the ABC-AP algorithm generates N solutions initially and chooses best half of bees as the initial solutions. Bees visiting the best food locations will be "employed bees" for these food locations and the rest of the bees become "onlookers." At the rest of the cycles, employed bees and any existing onlooker bees visit the food locations found by the employed bees together. A bee finding the best fitness in the neighborhood will be an employed bee in the next cycle. By doing so, the greedy selection process is performed once in each cycle. The other deviation is that the employed bee associated with the best solution is spared from becoming scout even if there is no improvement in the nectar after the LIMIT number of cycles.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

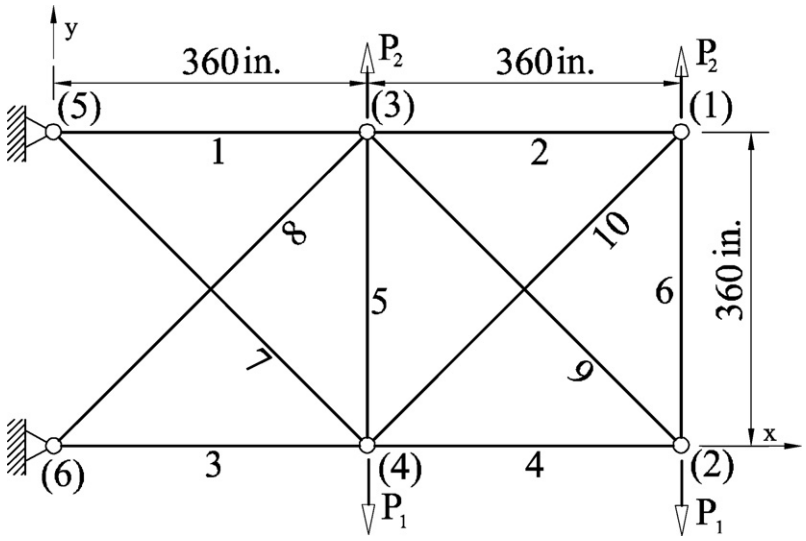

Fig. 1. Geometry and applied loading for a planar ten-bar truss.

7. Numerical examples

The following five classical test problems were used to investigate the numerical correctness, efficiency and validation of the ABC algorithm:

- A ten-bar plane structure subjected to two loading cases,
- An eighteen-bar plane structure subjected to one loading condition,
- A twenty-two-bar element space structure subjected to two loading conditions,
- A seventy-two-bar element space structure subjected to two loading conditions,
- A two-hundred-bar element planer structure subjected to three loading conditions.

Based on the algorithm presented in Section 6, a C# program was developed to design truss structures using object-oriented paradigm. The program is composed of five main classes, namely DataProcess, Bee, LinearAnalyis, Matrix, IndataData. The class InputData reads input file and creates the container lists that store information concerning elements, nodes, loads and material descriptions. The class Matrix is used to solve the simultaneous algebraic equations and to carry out the matrix related operations. The class LinearAnalyis uses the class Indata and the class Matrix to perform the structural analysis procedure to determine the nodal displacements and element forces based on the displacement based finite element procedure. The class Bee was created to store the data such as fitness, penalty coefficient, name, design variables and actions related to the bee behaviors such as finding the food source. The class DataProcess combines all classes to perform the optimization procedure.

The optimization software was run on a Personal Computer with a Pentium Dual Core 2.33 GHz processor and 2 GB memory under the Microsoft Windows XP operating system.

For all examples presented in this study, the ABC-AP algorithm parameters were set as follows: a colony of bee size $N = 50$, the maximum number of cycles MNC = 1000D (D is the number of design variables) and LIMIT = MNC/3. Ten independent runs were performed with the best and worst performances being presented for each problem.

7.1. Ten-bar plane truss

Fig. 1 shows the geometry and the loading condition of the cantilever truss structures consisting of ten bars. This structure is a standard example used by many researchers including Lee and Geem [6], Hatay and Toklu [15], Li et al. [23], Perez and Bendinan [24], Fleury and Schmit [42], Renwei and Peng [43], Dobbs and Nelson [44]. In this example, two loading cases were considered: Case I in which the single loading condition $P_1 = 100$ kips and $P_2 = 0$ was considered and Case II in which the single loading condition $P_1 = 150$ kips and $P_2 = 50$ kips was considered. All members were assumed to be made from a material with an elastic modulus of $E = 10,000$ ksi and a mass density of $\rho = 0.10$ lb/in.^{3}. The cross-sectional areas of all members were included as sizing variables. The minimum and maximum cross-sectional areas of members were set to $A^l = 0.1$ in.^{2} and $A^u = 35.0$ in.^{2}. The displacements of the free nodes in both directions had to be less than ± 2 in and the allowable stress was set to ± 25 ksi. The problem had 32 nonlinear constraints (10 tension constraints, 10 compression constraints and 12 displacement constraints).

To study the effect of the colony size on the convergence rate of the ABC-AP algorithm, five different colonies consisting of 10, 30 50, 60 and 100 bees were used. The averages of each set of 10 independent runs for each colony are given in Fig. 2 where the objective function versus cycle numbers is shown. It can be seen from this figure that the convergence rates increase with greater numbers of bees. After 1000 cycles, with the exception of the colony of 10 bees the results of all the colonies are very close to each other and are the almost same after 15,000 cycles. The colony size may be set at any value between 30 and 100. In current research the colony size was set at 50 bees for all examples. Even if the maximum number of cycle is limited to any value between 1000 and 15,000, Fig. 2 shows that the results continue to improve while the number of cycles increases. Comparing the results of the ABC-AP with algorithms in the literature, the maximum number of cycles was set at 1000D (D is the number of design variables) for all examples. The results of the average of 10 independent runs for the original ABC algorithm [32] and the proposed ABC-AP algorithm are presented in Fig. 3. The results of the ABC-AP algorithm are significantly better than the ABC algorithm for the first 1000 cycles. The difference between the results of two algorithms becomes less after 1000 cycles but the ABC-AP always gives better results than the ABC algorithm.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

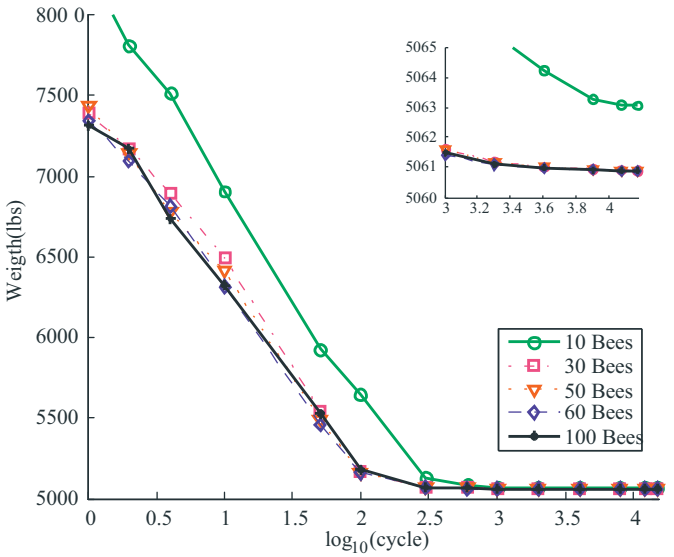

Fig. 2. Comparison of the convergence rates of five different colonies of 10, 30, 50, 60 and 100 bees for a ten-bar truss subjected to load Case I.

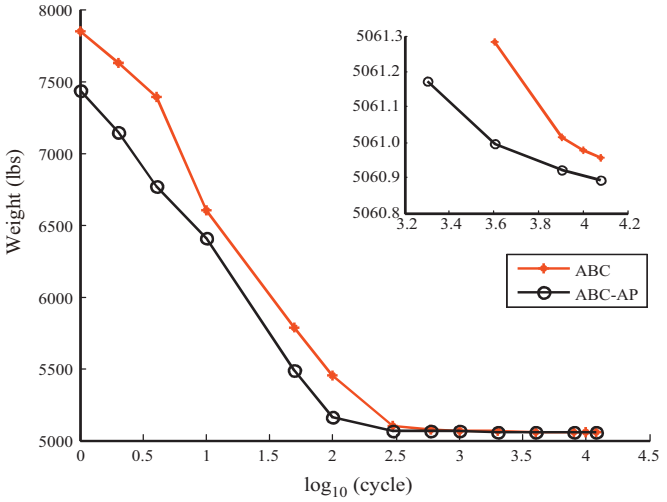

Fig. 3. Comparison of the results of original ABC and proposed ABC-AP algorithm for a ten-bar truss subjected to the load Case I.

Tables 1 and 2 show the minimal and maximal results of ten independent runs for a ten-bar truss subjected to Case I and Case II loads, respectively. Other published results found for the same problem using different optimization approaches including the optimality criteria [43,44], the simulated annealing (SA) algorithm [15], the heuristic practical swarm optimization (HPSO) algorithm [23], the particle swarm optimization (PSO) algorithm [24], and the harmony search (HS) algorithm [6] are also listed in Tables 1 and 2. The minimum weights obtained from the ABC-AP are 5060.880 lbs and 4677.077 lbs for Case I and Case II, respectively. There is no

Table 1
Optimization results for a ten-bar truss (Case I).

Variables		Optimal cross section area (in.^{2})						
No	Design name	Renwei and Peng [43]	Hatay and Toklu [15]	Li et al. [23]	Lee and Geem [6]	Perez and Behdinan [24]	This study	
							Best	Worst
1	A_1	30.590	30.680	30.704	30.150	33.500	30.548	30.465
2	A_2	0.100	0.100	0.100	0.102	0.100	0.100	0.100
3	A_3	23.270	23.500	23.167	22.710	22.766	23.180	23.109
4	A_4	15.190	14.970	15.183	15.270	14.417	15.218	15.184
5	A_5	0.100	0.100	0.100	0.102	0.100	0.100	0.100
6	A_6	0.460	0.550	0.551	0.544	0.100	0.551	0.556
7	A_7	7.500	7.450	7.460	7.541	7.534	7.463	7.477
8	A_8	21.070	21.020	20.978	21.560	20.467	21.058	21.154
9	A_9	21.480	21.430	21.508	21.450	20.392	21.501	21.521
10	A_{10}	0.100	0.100	0.100	0.100	0.100	0.100	0.100
Weight (lb)		5062.170	5061.600	5060.920	5057.880	5024.210	5060.880	5060.948
Constraint violation		None	None	None	0.907×10^{-3}	23.95×10^{-3}	None	0.358×10^{-6}

Note: 1 in.^{2} = 6.452 cm^2; 1lb = 4.45 N.

Table 2
Optimization results for a ten-bar truss (Case II).

Variable		Optimal cross section area (in.^{2})				
No	Design name	Dobbs and Nelson [44]	Li et al. [23]	Lee and Geem [6]	This study	
					Best	Worst
1	A_1	25.810	23.353	23.250	23.4692	23.5759
2	A_2	0.100	0.100	0.102	0.1005	0.1014
3	A_3	27.230	25.502	25.730	25.2393	25.2051
4	A_4	16.650	14.250	14.510	14.3540	14.3017
5	A_5	0.100	0.100	0.100	0.1001	0.1003
6	A_6	2.024	1.972	1.977	1.9701	1.9704
7	A_7	12.780	12.363	12.210	12.4128	12.4162
8	A_8	14.220	12.894	12.610	12.8925	12.8585
9	A_9	22.140	20.356	20.360	20.3343	20.3541
10	A_{10}	0.100	0.101	0.100	0.1000	0.1001
Weight (lb)		5059.70	4677.290	4668.810	4677.077	4677.306
Constraint violation		None	25.0×10^{-6}	3.561×10^{-3}	None	None

Note: 1 in.^{2} = 6.452 cm^2; 1lb = 4.45 N.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

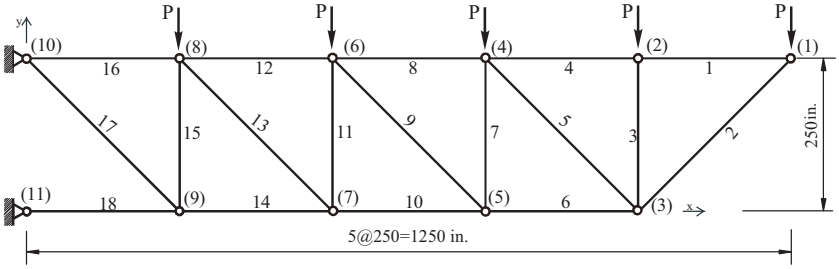

Fig. 4. Geometry and applied loading for a planar eighteen-bar truss.

constraint violation for either case. The ABC-AP algorithms provide very good results when compared with other results, for example, those obtained from the HS algorithm [6] and the PSO algorithm [24] are lighter than the results from the ABC-AP but the normalized constraint violations for Case I are approximately 0.907×10^{-3} and 23.95×10^{-3}, respectively. The SA, the PSO, the HS and the HPSO algorithms complete the optimization process after 1000×10^3, 5×10^3, 400×10^3 and 75×10^3 function evaluations, respectively. The ABC-AP algorithm performs 500×10^3 function evaluations and it took approximately 11 s to complete the process. It is interesting to note that the difference between minimal and maximal results obtained from the ABC-AP is less than 0.001 % for both loading cases. On the other hand this difference is about 3% for PSO algorithm.

7.2. Eighteen-bar plane truss

Fig. 3 shows the geometry and loading condition of the cantilever truss structures consisting of eighteen bars and 11 nodes. This problem has previously been presented as an example of sizing and layout optimization by several researchers however, only size optimization is considered in this work. The structure was subjected to a single loading condition which is a series of concentrated point loads of 20 kips acting on the upper cord nodes of the truss as in Fig. 4. All members were assumed to be constructed from material with an elastic modulus of $E = 10,000$ ksi and a mass density of $\rho = 0.10$ lb/in.^{3}. The stress constraint is defined as 20 ksi for both the tension and compression members. In addition, the Euler buckling constraint is also taken into account for compression members. The Euler buckling stress for the ith member is calculated as:

$$\sigma_i = -\frac{KEA_i}{L_i^2} \tag{11}$$

where L_i and A_i are the length and the cross-section area of the member, respectively. 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 follows: (i) elements 1, 4, 8, 12 and 16; (ii) elements 2, 6, 10, 14 and 18; (iii) elements 3, 7, 11 and 15; (iv) elements 5, 9, 13 and 17. The minimum cross-sectional area of the members was $A^l = 0.10$ in.^{2} and the maximum cross-section was set to 50 in.^{2}, even if there is no maximum cross-section limitation. There were 36 nonlinear constraints on the member stress and buckling stress with no displacement constraints.

The minimal and maximal results from 10 independent runs of the ABC-AP algorithm are presented Table 3. In addition, this table contains the results for the same optimization task from different research efforts including Imai and Schmit [45] who used the multiplier method and Lee and Geem [6] who used the HS algorithm to solve the problem. The results of the ABC-PA (6340.529 lbs) are almost same as the results from Imai and Schmit (6340 lbs). Although, the HS algorithm [6] produced a lighter design (6421.88 lbs) than the algorithm presented in this study (6340.529 lbs), the algorithm violates the stress constraints which were approximately 7.508×10^{-3} while the optimal solution found by the ABC-AP meets all constraint requirements. The HS algorithm completes the optimization process after 400×10^3 structural analysis. The HS algorithm found an optimum weight after less than 60 s while the ABC-AP algorithm required 200×10^3 structural analysis that took about 6 s to complete the optimization process.

7.3. Twenty-five-bar space truss

The space 25-bar truss shown in Fig. 5 has been optimized by several researchers including Lee and Geem [6], Lamberti [14], Fleury and Schmit [42], Haftka and Gurdal [7]. All members were assumed to be made of a material with an elastic modulus of $E = 10,000$ ksi and a mass density of $\rho = 0.01$ lb/in.^{3}. This space truss was subjected to two loading conditions as shown in Table 4. The struc-

Table 4
Nodal loading components (kips) for a twenty-five-bar truss.

Loading cases	Node	x	y	z
1	1	1.000	10.000	−5.000
	2	0	10.000	−5.000
	3	0.500	0	0
2	5	0	20.000	−5.000
	6	0	−20.000	−5.000

Note: 1 kips = 4.45 kN.

Table 3
Optimization results for an eighteen-bar truss structure.

Variables		Optimal cross section area (in.^{2})			
No	Design variables	Imai and Schmit [45]	Lee and Geem [6]	This study	
				Best	Worst
1	$A_1, A_4, A_8, A_{12}, A_{16}$	9.998	9.980	10.000	10.000
2	$A_2, A_6, A_{10}, A_{14}, A_{18}$	21.650	21.630	21.651	21.651
3	A_3, A_7, A_{11}, A_{15}	12.500	12.490	12.500	12.500
4	A_5, A_9, A_{13}, A_{17}	7.072	7.057	7.071	7.071
Weight (lb)		6430.000	6421.880	6430.529	6430.529
Constraint violation		0.259×10^{-3}	7.508×10^{-3}	None	None

Note: 1 in.^{2} = 6.452 cm^2; 1lb = 4.45 N.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

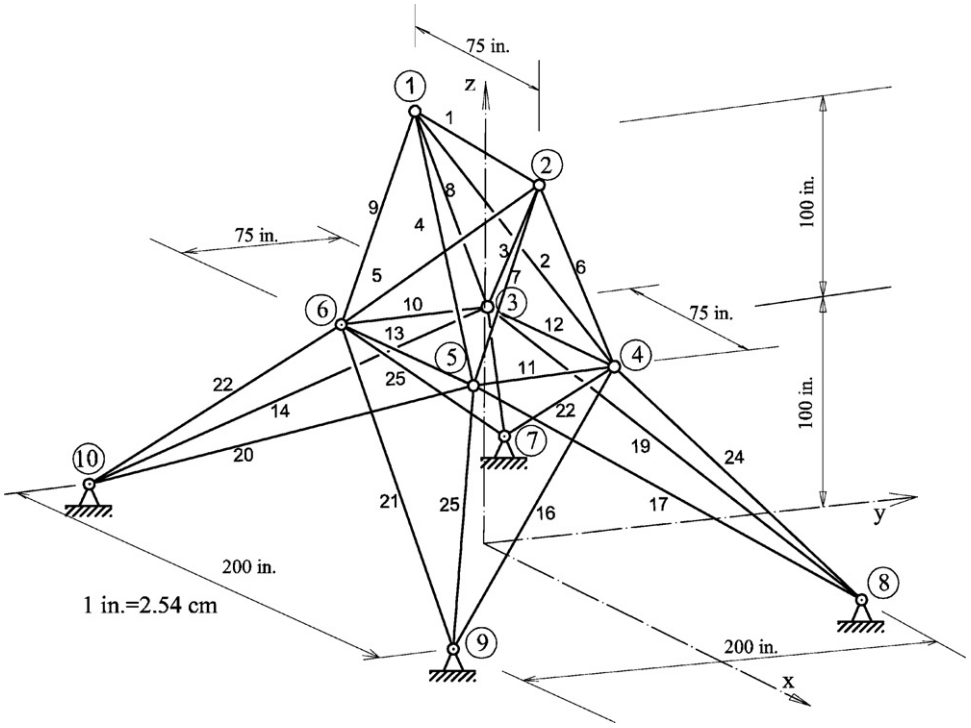

Fig. 5. Geometry of a twenty-five-bar truss structure.

tural analysis program was run twice for each distinct loading condition and the unconstrained fitness function was calculated based on the corresponding stress and displacement constraint violations. Since the structure was doubly symmetric about the *x*- and *y*-axes, the problem involves eight independent design variables after linking in order to impose symmetry. The members were grouped as follows: (1) element 1; (2) elements 2, 3, 4 and 5; (3) elements 6, 7, 8 and 9; (4) elements 10 and 11; (5) elements 12 and 13; (6) elements 14, 15, 16, 17 and 18; (7) elements 18, 19, 20 and 21; (8) elements 22, 23, 24 and 25. The minimum cross-sectional area of each member was set to 0.01 in.^{2}. The displacements at the top nodes 1 and 2 in all directions had to be less than ±0.35. The allowable stresses for all members are given in Table 5. The problem had 124 nonlinear constraints (25 tension, 25 compression, 6 positive displacement and 6 negative displacements for each loading condition).

Table 6 shows the best and worst results of the ABC-AP algorithm with 8 size variables and compares these results with those previously reported in the literature. The difference between all the results is less than 0.006%, so all optimization algorithm listed in Table 6 found almost the same structural weight. Constraint violation (0.0122) occurred is only for the HS algorithm [6]. The HS, the HPSO [23] and the Corrected Multi-Level & Multi-Point Simulated Annealing (CMLPSA) [14] algorithms required 300×10^3, 7125 and 400 structural analysis to complete the search process while the ABC-AP found the optimum weight of 545.1927 lbs after

Table 5
Allowable stresses (ksi) for a twenty-five-bar truss.

Design variables	Members	Compression	Tension
1	A_1	35.092	40.000
2	A_2–A_5	11.590	40.000
3	A_6–A_9	17.305	40.000
4	A_{10}–A_{11}	35.092	40.000
5	A_{12}–A_{13}	35.092	40.000
6	A_{14}–A_{17}	6.759	40.000
7	A_{18}–A_{21}	6.959	40.000
8	A_{22}–A_{25}	11.082	40.000

Note: 1 ksi = 6.89725 MPa.

Table 6
Optimization results for a twenty-five-bar truss structure.

Variables		Optimal cross section area (in.^{2})					
No	Design variables	Haftka and Gtirdal [7]	Lee and Geem [6]	Li et al. [23]	Lamberti [14]	This study	
						Best	Worst
1		0.010	0.047	0.010	0.010	0.011	0.010
2	A_2–A_5	1.987	2.022	1.970	1.987	1.979	2.006
3	A_6–A_9	2.991	2.950	3.016	2.994	3.003	2.961
4	A_{10}–A_{11}	0.010	0.010	0.010	0.010	0.010	0.012
5	A_{12}–A_{13}	0.012	0.014	0.010	0.010	0.010	0.010
6	A_{14}–A_{17}	0.683	0.688	0.694	0.694	0.690	0.689
7	A_{18}–A_{21}	1.679	1.657	1.681	1.681	1.679	1.677
8	A_{22}–A_{25}	2.664	2.663	2.643	2.643	2.652	2.665
Weight (lb)		545.220	544.380	545.190	545.161	545.193	545.276
Constraint violation		None	0.0122	None	None	None	None

Note: 1 in.^{2} = 6.452 cm^2; 1lb = 4.45 N.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

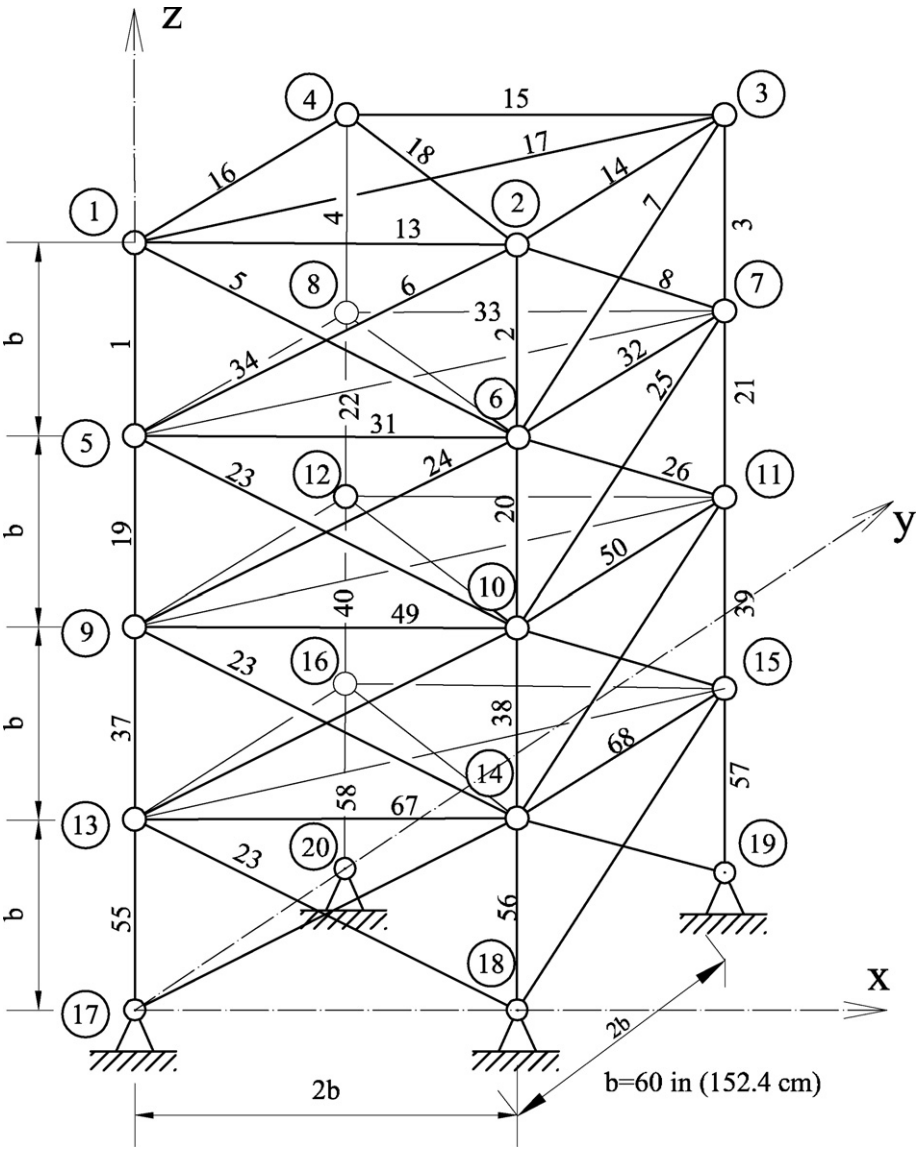

Fig. 6. Geometry of a seventy-two-bar truss.

approximately 300×10^3 cycles, which took approximately 32 s. The CMLPSA algorithm which is modified version of the simulated Annealing algorithm requires the gradient of the objective function; therefore, the number of cycles performed by CMLPA [14] to find the optimal design was less than the other direct search algorithms.

7.4. Seventy-two-bar truss

The 72-bar four level skeletal tower is shown in Fig. 6 and the truss was subjected to two loading conditions as given in Table 7.

Table 7
Nodal loading components (kips) for a twenty-five-bar truss.

Loading conditions	Node	*x*	*y*	*z*
1	1	5.000	5.000	−5.000
2	1	0	0	−5.000
	2	0	0	−5.000
	3	0	0	−5.000
	4	0	0	−5.000

Note: 1 kips = 4.45 kN.

The minimum cross-sectional area of each member was set to 0.01 in.^{2}. The displacements at the upper most nodes 1, 2, 3 and 4 in *x*-direction and *y*-direction had to be less than ±0.25 in. The allowable stresses for all members were ±25 ksi. The problem had 320 nonlinear constraints (72 tension, 72 compression, 8 positive displacements, 8 negative displacements for each loading case). Due to the symmetry of the structure, 16 independent design variables were used for linking. The member numbers and the corresponding group numbers are given in Table 8.

Table 8
Element grouping for the seventy-two-bar truss structures.

Group	Elements	Group	Elements
1	1, 2, 3,4	9	37, 38, 39, 40
2	5, 6, 7, 8, 9, 10, 11, 12	10	41, 42, 43, 44, 45, 46, 47, 48
3	13, 14, 15, 16	11	49, 50, 51, 52
4	17, 18	12	53, 54
5	19, 20, 21, 22	13	55, 56, 57, 58
6	23, 24, 25, 26, 27	14	59, 60, 61, 62, 63, 64, 65, 66
7	31, 32, 33, 34	15	67, 68, 69, 70
8	35, 36	16	71, 72

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

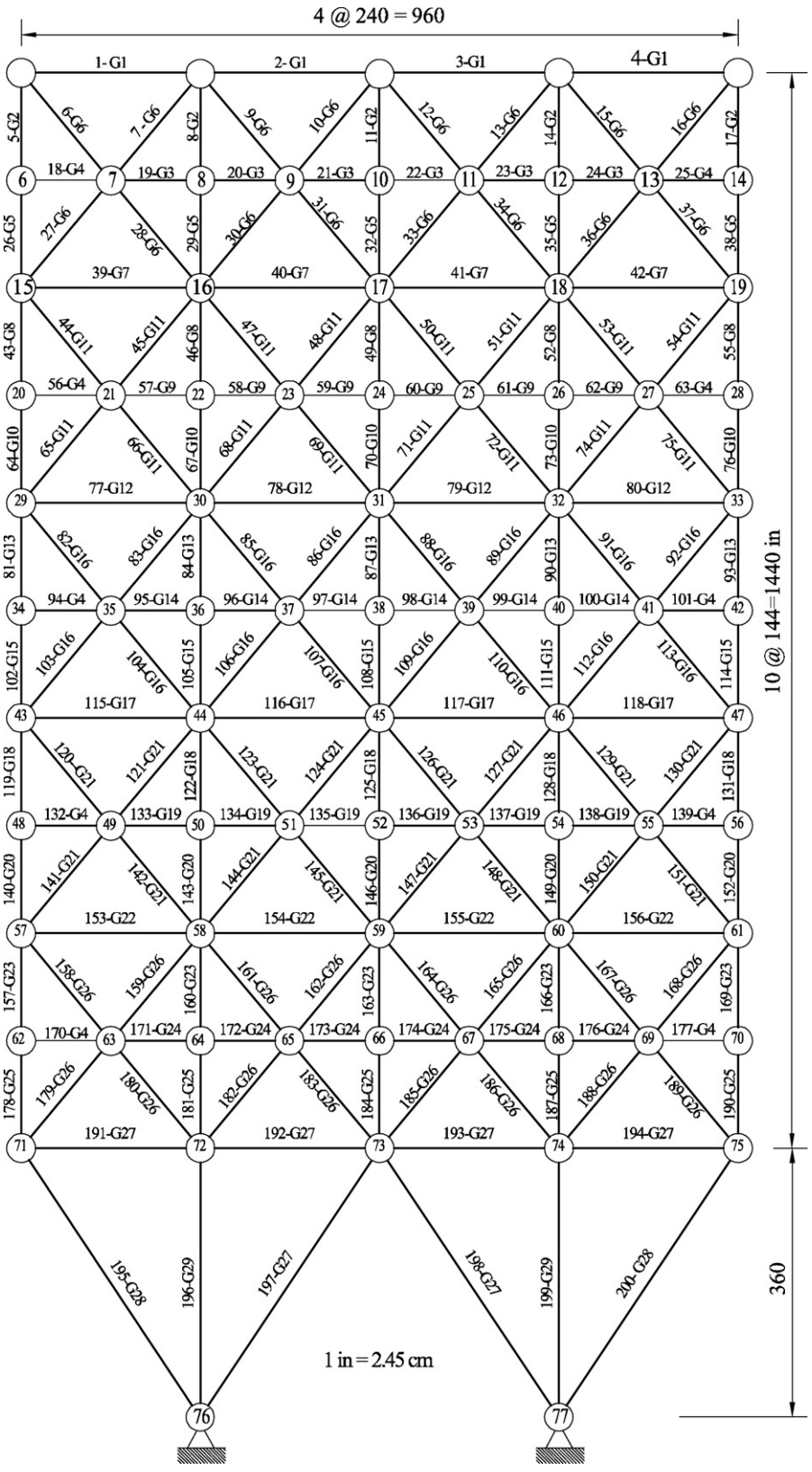

Fig. 7. Geometry, Node numbers, member numbers and group numbers of a two-hundred-bar truss.

The ABC-AP algorithm found the optimal weight of the seventy-two-bar truss to be 363.8392 lbs and there were no constraint violations after 400×10^3 structural analysis, which took 228 s. Optimal designs reported in the literature together with the ABC-AP algorithm are listed in Table 9. The optimum design weight obtained from the CMLPSA [14], the HS [6] and the HPSO [23] algorithms are almost same as that of the ABC-AP algorithm. These three algorithms exhibit some amount of constraint violation but the ABC-AP algorithm did not produce any constraint violation. In this sense, the ABC-AP algorithm outperforms the other algorithms.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

Table 9
Optimization results for a seventy-two-bar truss structure.

Variables		Optimal cross section area (in.^{2})						
No	Design variables	Erbatur et al. [9]	Perez and Behdinan [24]	Li et al. [23]	Lee and Geem [6]	Lamberti [14]	This study	
							Best	Worst
1	A_1–A_4	0.155	0.1615	1.907	1.963	0.1665	0.1675	0.1683
2	A_5–A_{12}	0.535	0.5092	0.524	0.481	0.5363	0.5346	0.5336
3	A_{13}–A_{16}	0.480	0.4967	0.010	0.010	0.4460	0.4443	0.4462
4	A_{17}–A_{18}	0.520	0.5619	0.010	0.011	0.5761	0.5803	0.5849
5	A_{19}–A_{22}	0.460	0.5142	1.288	1.233	0.5207	0.5208	0.5140
6	A_{23}–A_{30}	0.530	0.5464	0.523	0.506	0.5180	0.5178	0.5217
7	A_{31}–A_{34}	0.120	0.1000	0.010	0.011	0.0100	0.0100	0.0100
8	A_{35}–A_{36}	0.165	0.1095	0.010	0.012	0.1141	0.1048	0.0973
9	A_{37}–A_{40}	1.155	1.3079	0.544	0.538	1.2903	1.2968	1.2847
10	A_{41}–A_{48}	0.585	0.5193	0.528	0.533	0.5170	0.5191	0.5138
11	A_{49}–A_{52}	0.100	0.1000	0.019	0.010	0.0100	0.0100	0.0100
12	A_{53}–A_{54}	0.100	0.1000	0.020	0.167	0.0100	0.0101	0.0100
13	A_{55}–A_{58}	1.755	1.7427	0.176	0.161	1.8866	1.8907	1.9010
14	A_{59}–A_{66}	0.505	0.5185	0.535	0.542	0.5169	0.5166	0.5211
15	A_{67}–A_{70}	0.105	0.1000	0.426	0.478	0.0100	0.0100	0.0100
16	A_{71}–A_{72}	0.155	0.1000	0.612	0.551	0.0100	0.0100	0.0100
Weight (lb)		385.760	381.91	364.86	364.330	363.803	363.8392	363.8683
Constraint violation		None	None	13.701	12.06	0.04×10^{-3}	None	None

Note: 1 in.^{2} = 6.452 cm^2; 1lb = 4.45 N.

7.5. Two-hundred bar planar truss

This 200-bar truss problem was previously studied by Lee and Geem [6] and Lamberti [14]. The members were linked together into twenty-nine groups. Fig. 7 shows the geometry, node numbering, material numbering and group numbers. The modulus of elasticity and the material density of all members were 30, 000 ksi and $\rho = 0.283$ lb/in.^{3}, respectively. The members were subjected to stress limitations of ± 10 ksi. There was no displacement limit but the minimum cross-section area was not allowed to be less than 0.1 in.^{2}. There were three loading conditions: (1) 1.0 kips acting in the positive x-direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62 and 71; (2) 10 kips acting in the negative y-direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 29, . . ., 72,

Table 10
Optimization results for a two-hundred-bar truss structures.

Variables	Optimal cross section area (in.^{2})					
Design variables	Lee and Geem [6]	Lamberti [14]	This study			
		$A = 10$ in.^{2}	After 350D cycles		After 1000D cycles	
			Best	Worst	Best	Worst
1	0.1253	0.1468	0.1042	0.1029	0.1039	0.1125
2	1.0157	0.9400	0.9416	0.9610	0.9463	0.9580
3	0.1069	0.1000	0.1030	0.1016	0.1037	0.1060
4	0.1096	0.1000	0.1118	0.1298	0.1126	0.1050
5	1.9369	1.9400	1.9430	1.9627	1.9520	1.9654
6	0.2686	0.2962	0.2945	0.2948	0.2930	0.2995
7	0.1042	0.1000	0.1072	0.1060	0.1064	0.1109
8	2.9731	3.1042	3.1301	3.1218	3.1249	3.1221
9	0.1309	0.1000	0.1361	0.1322	0.1077	0.1029
10	4.1831	4.1042	4.1856	4.1144	4.1286	4.1472
11	0.3967	0.4034	0.4313	0.4627	0.4250	0.4343
12	0.4416	0.1912	0.1073	0.1475	0.1046	0.1482
13	5.1873	5.4284	5.4680	5.5498	5.4803	5.4855
14	0.1912	0.1000	0.1495	0.1160	0.1060	0.1192
15	6.2410	6.4284	6.4876	6.5246	6.4853	6.4813
16	0.6994	0.5734	0.5782	0.6331	0.5600	0.5913
17	0.1158	0.1327	0.2031	0.2214	0.1825	0.1924
18	7.7643	7.9717	8.0759	8.1406	8.0445	8.0633
19	0.1000	0.1000	0.2015	0.1882	0.1026	0.1149
20	8.8279	8.9717	9.0666	9.2081	9.0334	9.1289
21	0.6986	0.7049	0.8548	0.8835	0.7844	0.8015
22	1.5563	0.4196	0.4106	0.5155	0.7506	0.9545
23	10.9806	10.8636	11.2225	11.7019	11.3057	11.5255
24	0.1317	0.1000	0.1840	0.2730	0.2208	0.4215
25	12.1492	11.8606	12.2790	12.4107	12.2730	12.4972
26	1.6373	1.0339	1.2040	1.3503	1.4055	1.6899
27	5.0032	6.6818	5.6580	5.1542	5.1600	4.5072
28	9.3545	10.8113	10.2616	9.9173	9.9930	9.4678
29	15.0919	13.8404	14.417	14.79213	14.70144	15.30332
Weight (lb)	25447.100	25447.528	25600.030	25832.780	25533.79	25756.640
Constraint violation	0.40023	0.00310	0.00036	0.00022	None	0.00220

Note: 1 in.^{2} = 6.452 cm^2; 1lb = 4.45 N.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

73, 74 and 75; (3) conditions 1 and 2 acting together. The problem had 1200 nonlinear constraints (200 tension and 200 compression stress constraints for each loading case).

Table 10 lists the designs developed by the ABC-AP algorithm, the HS algorithm [6] and the SA algorithm [14]. The best ABC-AP design results in a truss structure that weighs 25, 540.58 lbs after 1000*D* cycles and 25600.30 lbs after 350*D* cycles, which is about 0.36% and 0.60% heavier than the designs presented by Lee and Geem [6] and Lamberti [14], respectively. The ABC-AP algorithm found the optimal weights of the two-hundred bar truss after 1000*D* cycles, which took 2688 s and after 350*D* cycles which took 927 s. The best optimum design of the ABC-AP algorithm does not violate the constraints after 1000*D* cycles (1450×10^3 function evaluations) and the constraint violation is about 0.00036 for 350*D* cycles (507×10^3 function evaluations). On the other hand, normalized constraint violations are about 0.40 for the HS algorithm [6] and 0.003 for the CMLPSA algorithm [14]. The CMLPSA and HS algorithm completed the optimization process after 9620 and 960×10^3 function evaluations, respectively.

8. Conclusion

The Artificial Bee Colony (ABC) algorithm, based on mimicking the food foraging behavior of honeybee swarms, is proposed as a method of solving the optimization problems of planar and space truss structures. An adaptive penalty (AP) function method was integrated into the algorithm to transform constrained optimization problems to unconstrained optimization problems. Optimization software based on the ABC-AP algorithm was coded in the C# programming language with using object-oriented technology. Five test problems were studied using this optimization program to show that the ABC-AP algorithm can be successfully applied to the optimization problems of the truss structures subjected to multiple loading conditions. The comparison of the results of the ABC-AP with those of other algorithms demonstrated that the ABC-AP algorithm provides results as good as or better than other algorithms and can be used effectively for solving such problems. The algorithm shows a remarkably robust performance with a 100% success rate. The difference between the minimal and maximal results for all examples is less than 1% (an average of approximately 0.16%).

The ABC-AP algorithm shows the positive performance in three aspects. The first one is the initial point independency and global nature of the algorithm. The second aspect is the characteristic of the adaptive penalty function coefficient which changes in the course of the optimization based on the feedback from previous process. The last aspect is that the algorithm does not require the evaluation of the gradients of objective and constraint functions. This makes the ABC-AP algorithm easy to implement in a structural analysis application. On the other hand, the ABC algorithm does not show any improvement in the speed of convergence in terms of the number of the structural analyses performed to obtain the best designs.

In the research described in this paper, a number of optimization problems of trusses with continuous sizing variable and fixed geometry were solved very effectively. Further tests are required to determine whether the ABC-AP algorithm can be employed to solve optimization problems of other structural types with discrete sizing and configuration variables.

Acknowledgment

Grateful thanks to Prof. L. Lamberti - The Polytechnic of Bari, Bari, Italy for providing additional information about the example problems.

References

[1] J.S. Arora, Optimization of structural and mechanical systems, in: J.S. Arora (Ed.), Introduction to Optimization, World Scientific Publishing Pte. Co., Singapore, 2007, pp. 1–7.
[2] J.S. Arora, Methods for discrete variable structural optimization, in: S.A. Burns (Ed.), Recent Advances in Optimal Structural Design, Technical Committee on Optimal Structural Design, ASCE, Reston, VA, 2002, pp. 1–40.
[3] C.M. Foley, M.S. Mark, S. Voss, Optimized design of fully and partially restrained steel frames using advanced analysis and object-oriented evolutionary computation, Technical Report, National Science Foundation, MUE-001-2001, pp. 1–6.
[4] D.N. Lagaros, M. Papadrakakis, G. Kokossalakis, Structural optimization using evolutionary algorithms, Computers and Structures 80 (2002) 571–589.
[5] K. Deb, S. Gulati, Design of truss-structures for minimum weight using genetic algorithms, Finite Element in Analysis and Design 37 (2001) 447–465.
[6] K.S. Lee, Z.W. Geem, A new structural optimization method based on the harmony search algorithm, Computers and Structures 82 (2004) 781–798.
[7] R.T. Haftka, Z. Gürdal, Elements of Structural Optimization, 3rd ed., Kluwer Academic Publishers, The Netherlands, 1992, pp. 245–250.
[8] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley Publishing Co., Inc., Massachusetts, 1989.
[9] F. Erbatur, O. Hasançebi, İ. Tütüncü, H. Kılıç, Optimum design of planar and space structures with genetic algorithms, Computers and Structures 75 (2000) 209–224.
[10] H. Adeli, N.T. Cheng, Concurrent genetic algorithms for optimization of large structures, Journal of Aerospace Engineering, ASCE (1994) 276–296.
[11] V. Togan, A.T. Daloglu, Optimization of 3D trusses with adaptive approaches in genetic algorithms, Engineering Structures 28 (2006) 1019–1027.
[12] S.O. Degertekin, M.P. Saka, H.S. Hayaloglu, Optimal load and resistance factor design of geometrically nonlinear steel space frames via tabu search and genetic algorithm, Engineering Structures, doi:10.1016/j.engstruct.2007.03.014.
[13] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science (2002) 671–680.
[14] L. Lamberti, An efficient simulated annealing algorithm for design optimization of truss structures, Computers and Structures 86 (2008) 1936–1953.
[15] T. Hatay, Y.C. Toklu, Optimization of truss using simulated annealing method, in: Fifth International Congress on Advances in Civil Engineering, Istanbul Technical University, Istanbul, Turkey, 25–27 September, 2002, pp. 379–388.
[16] O. Hasancebi, F. Erbatur, On efficient use of simulated annealing in complex structural optimization problems, Acta Mechanics 157 (2002) 27–50.
[17] R. Kicinger, T. Arciszewki, K.D. Jong, Evolutionary computation and structural design: a survey of the state-of-the-art, Computers and Structures 83 (2005) 1943–1978.
[18] O. Hasancebi, Adaptive evolution strategies in structural optimization: enhancing their computational performance with applications to large-scale structures, Computers and Structures 87 (2008) 119–132.
[19] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford Press, 1999.
[20] E. Bonabeau, C. Meyer, Swarm intelligence: a whole new way to think about business, Harvard Business Review, R0105G (2001) 106–114.
[21] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, vol. 4, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[22] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, USA, 2001.
[23] L.J. Li, Z.B. Huang, F. Liu, Q.H. Wu, A heuristic particle swarm optimizer for optimization of pin connected structures, Computers and Structures 85 (2007) 340–349.
[24] R.E. Perez, K. Behdinan, Particle swarm intelligence for structural design optimization, Computers and Structures 85 (2007) 1579–1588.
[25] T. Stautzle, M. Dorigo, ACO Algorithms for the traveling salesman problem, in: K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux (Eds.), Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications, John Wiley & Sons, 1999.
[26] C. Camp, B. Bichon, S.P. Stovall, Design of steel frames using ant colony optimization, Journal of Structural Engineering 131 (3) (2005) 369–379.
[27] D. Teodorovic, Transport modeling by multi-agent systems: a swarm intelligence approach, Transportation Planning and Technology 26 (4) (2003) 289–312.
[28] D. Teodorovic, M.D. Orco, Bee colony optimization—a comparative learning approach to computer transportation problems, Advanced or an IA Methods in Transportation (2005) 51–60.
[29] X. Yang, Engineering optimization via Nature-Inspired Virtual Bee Algorithms, in: J. Mira, J.R. Alvarez (Eds.), IWINAC 2005, LNCS 3562, 2005, pp. 317–323.
[30] D.T. Pham, A. Granbarzadeh, E. Koç, S. Otri, S. Rahim, M. Zaidi, The bees algorithm – a novel tool for complex optimization problems, in: Innovation Production Machines and System Virtual Conference, 2006, http://conference.iproms.org.
[31] D.T. Pham, E. Koç, A. Granbarzadeh, S. Otri, Optimisation of the weight of multi-layered perceptrons using the bees algorithm, in: Proceedings of 5th International Symposium of Intelligence Manufacturing Systems, Sakarya, Turkey, 2006, pp. 38–46.
[32] D. Karaboga, B. Basturk, On the performance of Artificial Bee Colony (ABC), Applied Soft Computing 8 (1) (2008) 687–697.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003

[33] D. Karaboga, B. Basturk, Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems, in: P. Melin, et al. (Eds.), IFSA 2007, LNAI 4529, 2007, pp. 789–798.
[34] B. Basturk, D. Karaboga, An Artificial Bee Colony (ABC) algorithm for numerical function optimization, in: IEEE, Swarm Intelligence Symposium, Indianapolis, IN, USA, 2006.
[35] A. Singh, An Artificial Bee Colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing 9 (2) (2009) 625–631.
[36] Honey Bee Biology, Texas A&M University, Department of Entomology, http://honeybee.tamu.edu/about/biology.html.
[37] Apiary Fact sheets, Ministry of Agriculture and Lands of British Colombia, http://www.al.gov.bc.ca.
[38] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: survey of the state of the art, Computer Methods in Applied Mechanics and Engineering 191 (2002) 1245–1287.
[39] A.B. Hadj-Alouane, J.C. Bean, A genetic algorithm for the multiple-choice integer program, Operation Research 45 (1997) 92–101.
[40] P. Nanakorn, K. Meesomklik, An adaptive penalty function in genetic algorithm for structural design optimization, Computers and Structures 78 (2001) 2527–2539.
[41] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering 186 (2–4) (2000) 311–338.
[42] C. Fleury, L.A. Schmit, Dual methods and approximation concepts in structural synthesis, NASA Contractor Report 3226 (1980) 97–124.
[43] X. Renwei, L. Peng, An efficient method for structural optimization, Acta Mechanica Sinica 2 (4) (1986) 348–361.
[44] M.W. Dobbs, R.B. Nelson, Application of optimality criteria to automated structural design, AIAA Journal 14 (10) (1976) 1436–1443.
[45] K. Imai, L.A. Schmit, Configuration optimization of trusses, Journal of Structural Division ASCE 107 (ST5) (1991) 745–756.

Mustafa Sonmez is an Assistant Professor in the Civil Engineering Department at Aksaray University, Turkey. He received his B.Sc. in Civil Engineering with honors from Middle East Technical University, Turkey in 1991 and his M.Sc. and Ph.D. in civil engineering (majoring in structural engineering) from University of Pittsburgh, Pennsylvania, USA in 1996 and 2000, respectively. Dr. Sonmez's research interests are in the areas of nonlinear analysis of frame and truss structures, computer programming and optimization algorithms. He has authored and co-authored refereed journal papers and numerous conference proceedings. In addition to conducting research, Dr. Sonmez teaches a variety of undergraduate and graduate courses at Aksaray University.

Please cite this article in press as: M. Sonmez, Artificial Bee Colony algorithm for optimization of truss structures, Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc.2010.09.003