J Intell Manuf (2013) 24:729–740
DOI 10.1007/s10845-011-0621-6

An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems

Ivona Brajevic · Milan Tuba

Received: 25 October 2011 / Accepted: 26 December 2011 / Published online: 10 January 2012

Abstract Artificial bee colony (ABC) algorithm developed by Karaboga is a nature inspired metaheuristic based on honey bee foraging behavior. It was successfully applied to continuous unconstrained optimization problems and later it was extended to constrained design problems as well. This paper introduces an upgraded artificial bee colony (UABC) algorithm for constrained optimization problems. Our UABC algorithm enhances fine-tuning characteristics of the modification rate parameter and employs modified scout bee phase of the ABC algorithm. This upgraded algorithm has been implemented and tested on standard engineering benchmark problems and the performance was compared to the performance of the latest Akay and Karaboga's ABC algorithm. Our numerical results show that the proposed UABC algorithm produces better or equal best and average solutions in less evaluations in all cases.

Keywords Artificial bee colony (ABC) · Constrained optimization · Swarm intelligence · Nature inspired metaheuristics

This research is supported by Ministry of Education and Science of Republic of Serbia, Grant No. III-44006.

I. Brajevic
Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade, Serbia
e-mail: ivona.brajevic@googlemail.com

M. Tuba (✉)
Faculty of Computer Science, Megatrend University Belgrade, Bulevar umetnosti 29, 11070 Belgrade, Serbia
e-mail: tuba@ieee.org

Introduction

Engineering problems normally include mixed, continuous and discrete, design variables, nonlinear objective functions and nonlinear constraints. Constraints are very important in these types of problems since they are usually imposed in the statement of the problem and sometimes are very hard to satisfy, which makes the search difficult and inefficient. General constrained optimization problem is to find x so as to

$$minimize\ f(x),\ x = (x_1, x_2, \dots, x_n) \in R^n \tag{1}$$

where $x \in F \subseteq S$. The objective function f is defined on the search space $S \subseteq R^n$ and $F \subseteq S$ defines the feasible region. The search space S is defined as an n-dimensional rectangle in R^n. The variable domains are limited by their lower and upper bounds:

$$l_i \le x_i \le u_i, \quad \text{for } i = 1, \dots, n \tag{2}$$

whereas the feasible region $F \subseteq S$ is defined by a set of m additional constraints ($m \ge 0$):

$$g_j(x) \le 0, \quad \text{for } j = 1, \dots, q \tag{3}$$
$$h_j(x) = 0, \quad \text{for } j = q + 1, \dots, m \tag{4}$$

Deterministic approaches to constrained nonlinear programming problems, such as sequential quadratic programming methods and generalized reduced gradient methods, are inflexible to adapt the solution algorithm to a given problem. Generally, a given problem is modeled in such a way that a classical algorithm can handle it. This usually requires making additional assumptions which might not be easy to justify. Therefore the applicability of such deterministic approaches is limited.

During last decades most approaches to hard constrained optimization problems are based on metaheuristics which make no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, they do not guarantee that the optimal solution is ever found.

Recently, nature inspired intelligence has become very popular. Some of the most famous nature inspired metaheuristics are those methods representing successful animal team behavior, such as swarm or flocking intelligence (Kennedy and Eberhart 1995), ant colonies (Dorigo and Gambardella 1997) or performance of bees. Pure versions of these algorithms are later enhanced to improve performance in general or for some class of the problems (Pan and Liu 2011; Wang et al. 2011; Jovanovic and Tuba 2011; Lu and Romanowski 2011). Sometimes they are combined (Aydin 2010; Jun-Qing Li and Xie 2010) or adjusted for multi-objective problems (Pasandideh et al. 2011; Cheshmehgaz et al. 2011). Swarm intelligence algorithms are successfully used for wide range of practical problems (Gaitonde and Karnik 2010; Puranik et al. 2011; Chang and Huang 2010).

The algorithms based on the behavior of bees are divided into three types according to their behavior in nature (Baykasoglu et al. 2007): the marriage behavior, the foraging behavior and the queen bee concept. Some of them were successfully applied to solve unconstrained and constrained optimization problems. The approach that simulates the marriage behavior of the bees is honey bees mating optimization algorithm (Haddad et al. 2006) proposed to solve highly non-linear constrained and unconstrained real valued mathematical models. The approaches that simulate the foraging behavior of bees and which were applied to continuous functional optimization problems are the virtual bee algorithm (Yang 2005), the bees algorithm (Pham et al. 2006) and the artificial bee colony (ABC) algorithm (Karaboga 2005).

The ABC algorithm was first designed for unconstrained optimization problems but was later also applied to constrained (engineering design) problems by extending the basic ABC algorithm simply by adding a constraint handling technique into the selection step of the ABC algorithm in order to prefer the feasible regions of the entire search space. In this paper an upgraded artificial bee colony (UABC) algorithm is presented. UABC algorithm enhances fine-tuning characteristics of the control parameters of the ABC algorithm in order to produce better balance of exploitation and exploration of candidate solutions in the search space. To show the power of this approach the UABC algorithm was applied to various standard benchmark engineering optimization problems and the obtained results were compared with the results reached by latest Akay and Karaboga's extended ABC algorithm (Akay and Karaboga 2010).

The rest of the paper is organized as follows. A brief literature review on ABC and its application are provided in section "Literature review". Our upgraded ABC algorithm is presented in section "Our proposed approach: UABC". The standard engineering design problems are described in section "Benchmark problems". Section "Experimental study" presents the adopted experimental setup and provides an analysis of the results obtained from our empirical study.

Literature review

Artificial bee colony algorithm proposed by Karaboga is a metaheuristic technique inspired by the foraging behavior of natural honey bee swarms (Karaboga 2005). In the ABC algorithm a colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts. All bees that are currently exploiting a food source are known as employed bees. Onlookers are those bees that are waiting in the hive for the employed bees to share information about the food sources presently being exploited by them. Employed bees share information about food sources by dancing in the dance area inside the hive. The dance is dependent on the nectar content of food source just exploited by the dancing bee. Onlooker bees watch the dance and choose a food source according to the probability proportional to the quality of that food source. Therefore, good food sources attract more onlooker bees. Scouts are those bees that are searching for new food sources in the neighborhood of the hive. The employed bee whose food source has been abandoned by the bees becomes a scout. Scout bees can be viewed as performing the job of exploration, whereas employed and onlooker bees can be viewed as performing the job of exploitation. Each food source is a possible solution for the problem and the nectar amount of a food source represents the quality of the solution represented by the fitness value. The fitness value of a solution x from Karaboga's ABC algorithm is:

$$fitness(x) = \begin{cases} 1/(1+f(x)), & \text{if } f(x) \geq 0 \\ 1+abs(f(x)), & \text{if } f(x) < 0 \end{cases} \tag{5}$$

The main steps of the algorithm are:

- send the employed bees onto the food sources and determine their nectar amounts,
- calculate the probability value of the sources by which they are preferred by the onlooker bees,
- stop the exploitation process of the sources abandoned by the bees,
- send the scouts into the search area for discovering new food sources, randomly
- memorize the best found source so far.

These steps are repeated the predefined number of times (maximum cycle number, MCN). In the basic ABC algorithm the control parameters are the size of the population which is equal to the sum of numbers of employed and onlooker bees (SP), the limit which represents the number of trials for releasing food source and the number of scouts.

The algorithm (Karaboga 2005) was tested on three well known test functions and it was concluded that the proposed algorithm can be used for solving uni-modal and multi-modal unconstrained numerical optimization problems.

Later Karaboga and Bastruck tested the performance of ABC algorithm for optimizing multivariable unconstrained functions (Karaboga and Basturk 2007b, 2008) and the results were compared with genetic algorithm (GA), particle swarm optimization (PSO) algorithm, and particle swarm inspired evolutionary algorithm (PS-EA) (Srinivasan and Seow 2003). The results showed that ABC outperforms the other algorithms.

Karaboga and Bastruck have also studied an extended version of the ABC for constrained optimization problems (Karaboga and Basturk 2007a). The performance of the algorithm, when it was applied to thirteen standard benchmark g functions, has been compared with the performance of the state-of-the-art algorithms based on the PSO and differential evolution (DE). It has been shown that ABC algorithm can be successfully used for solving constrained optimization problems. In that extended version of the ABC for constrained optimization, there are two additional control parameters in comparison with the basic version of the ABC. The first additional control parameter is the modification rate (*MR*) that controls the possible modifications of optimization parameters. The second additional control parameter is scout production period (*SPP*) that is used in scout bee phase of the algorithm and it denotes a predetermined period of cycles for producing artificial scouts. At every *SPP* cycle, if there is an abandoned food source, it is replaced with a new randomly produced solution. In order to handle the constraints of the problem, the extended ABC algorithm employs Deb's rules (Deb 2000). The method uses a tournament selection operator, where two solutions are compared at a time by applying the following criteria:

- Any feasible solution satisfying all constraints is preferred to any infeasible solution violating any of the constraints
- Among two feasible solutions, the one having better fitness value is preferred
- If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred, where the sum of constraint violation is defined as:

$$CV(x) = \sum_{j=1}^{q} max(0, g_j(x)) + \sum_{j=q+1}^{m} h_j(x) \qquad (6)$$

Akay and Karaboga proposed a version of the ABC algorithm to solve engineering design problems (Akay and Karaboga 2010). The difference from the version of ABC extended for constrained problems (Karaboga and Basturk 2007a) is in calculating probability values for solutions. In the version of ABC algorithm proposed in Akay and Karaboga (2010), probability values of infeasible solutions are between 0 and 0.5, while those of feasible solutions are between 0.5 and 1. In onlooker bee phase of the algorithm, by a selection mechanism, infeasible solutions are selected inversely proportional to their violation values. In the case of feasible solutions, they are selected probabilistically proportional to their fitness values. In the version of ABC algorithm for constrained problems, probability values for feasible solutions, as well as for infeasible solutions, are between 0 and 1, and they are are selected probabilistically proportional to their fitness values. The performance of the ABC algorithm when it was applied on five standard engineering problems was compared to the performance of society and civilization algorithm (SCA) (Ray and Liew 2003), versions of the PSO algorithm (He et al. 2004; Parsopoulos and Vrahatis 2005), and the evolution strategy ($((\mu + \lambda) - ES)$) (Mezura-Montes and Coello Coello 2005). The $((\mu+\lambda)-ES)$ achieved better performance in the terms of best solution, while the performance of ABC showed more robustness than the other algorithms. It was concluded that ABC is a promising tool for optimizing constrained engineering problems.

A new modification of the ABC algorithm for constrained optimization is proposed by Karaboga and Akay (2011). The difference from the version of the ABC algorithm proposed for engineering problems is in employed and onlooker bee phases of the algorithm as follows: even if MR parameter dictates that none of the solution parameters should be modified, one solution parameter is randomly selected and modified. This version is tested to solve the same thirteen benchmark problems as the version of ABC from Karaboga and Basturk (2007a). Its performance was compared with the performance of the state-of-the-art algorithms: homomorphous mapping (HM) method (Koziel and Michalewicz 1999), stochastic ranking (SR) method (Runarsson and Yao 2000), improved stochastic ranking (ISR) method and over-penalty (OPA) approach (Runarsson and Yao 2005), adaptive segregational constraint handling evolutionary algorithm (ASCEA) (Hamida and Schoenauer 2002), GA, simple multimembered evolution (SMES) (Mezura-Montes and Miranda-Varela 2005), PESO version of PSO (Zavala et al. 2005) and DE (latest results on DE are in Mezura-Montes et al. (2010)). Their computational results showed that the modified ABC algorithm can be efficiently used for solving constrained optimization problems.

Our proposed algorithm: UABC

In the population-based optimization algorithms both exploitation and exploration abilities are necessary. The exploitation refers to the ability of algorithm to apply the knowledge of the previous good solutions to find better solutions (by improving on these known good solutions). The exploration refers to the ability of algorithm to investigate the various unknown and less promising regions of the solution space to bring diversity and ability to avoid being trapped in local optima and eventually to discover the global optimum. The exploitation and exploration contradict each other and the two abilities should be well balanced in order to achieve good optimization performance.

Even though in general exploitation sticks to good known solutions, while exploration brings diversity, fine modifications and tuning of some population-based algorithm lead to more complex situations where it is possible (and desirable) to increase diversity during exploitation, or reduce diversity during exploration. Such modifications can improve algorithm performance since increasing diversity by some modification, while decreasing it by some other modification at the same stage, e.g. exploitation, may lead search to more desirable places, rather than canceling each other.

In our UABC algorithm in the onlooker phase (exploitation) we reduce the diversity, as may be expected, by reducing the space where new generated solution can be. In the original ABC algorithm (Akay and Karaboga 2010) new solution is generated from the current solution and one random solution so that new solution is inside the hyper-cube where these two old solutions are located at the opposing points of the cube's diagonal, while in our UABC algorithm only points on the diagonal, not from the whole cube, are permitted as new solutions.

On the other hand, original ABC algorithm lacks some diversity (or courage to search towards infeasible points) because in order to handle the constraints of the problem it employs Deb's rules and consequently, infeasible points are heavily suppressed. Additionally, in the Akay and Karaboga (2010) ABC algorithm, pseudo-code step 7, probability for selection by onlookers is calculated in such a way that all feasible solutions have that probability above 0.5 and all infeasible solutions below 0.5. Our UABC algorithm, like the ABC algorithm, employs Deb's rules, but we found that algorithm works better if mentioned discrimination is removed and infeasible solutions get better chance of being selected by being judged only on their fitness value (our pseudo-code step 8), not inversely proportional to their constraint violation values. This increases diversity in onlooker phase, but only in that specific direction of less discriminating infeasible solutions, which in no way contradicts previously mentioned reduced diversity by allowing new solutions only on the diagonal of the hyper-cube; just the opposite: both modifications work together towards faster convergence.

By doing such fine adjustments at the same stage, increasing and decreasing diversity, but differently in targeted directions, we better control the balance of exploitation and exploration.

In order to produce better solutions than Akay and Karaboga (2010) ABC algorithm proposed for engineering problems, our upgraded artificial bee colony (UABC) algorithm also enhances fine-tuning characteristics of modification rate (MR) parameter and employs modified scout bee phase. In the mentioned version of the ABC algorithm proposed for constrained problems and in the modified ABC algorithm for constrained problems (Karaboga and Akay 2011) there are several control parameters: size of population (SP), maximum cycle number (MCN), modification rate (MR), limit and scout production period (SPP). In the UABC algorithm we kept the control parameters SP, MCN and limit, but we divided MR parameter into two parameters: modification rate for employees (MRE) and modification rate for onlookers (MRO). Additionally, we included a new parameter which we named the infeasible solution replacement period ($ISRP$).

In order to produce neighborhood solutions our UABC algorithm uses MRE parameter in employed bee phase and MRO parameter in onlooker bee phase. Both parameters take values between 0 and 1. As the values for these parameters become higher, the probability of changing the optimization parameter x_j, $j = 1, 2, \dots, D$ is higher. It can be considered that if any of these two parameters take the smaller value we are looking for a new solution in the closer neighborhood. Otherwise, if any of the two parameters take the higher value we are looking for a new solution in the further neighborhood. Our experiments show that it is beneficial to amplify exploitation by staying in the closer neighborhood in the onlooker phase, so MRO parameter is set to a lower value.

In the scout bee phase we need to determine the food sources which can be abandoned, i.e. which are not worth exploiting anymore. Hence, the algorithm calculates values $failure_i$, $i = 1, 2, \dots, SP/2$, which represent the non-improvement number of the solution x_i used for the abandonment. We considered that solution can be abandoned if its $failure_i$ value is greater or equal to the value of limit control parameter. In the UABC algorithm, by the new parameter $ISRP$, the scout bee phase is changed as follows: after each $ISRP^{\text{th}}$ iteration, every infeasible solution and one feasible solution with the highest failure value that exceeds abandonment threshold (if such solutions exist) are replaced, each with a new randomly produced solution. In any other iteration only one solution (infeasible or feasible) with the highest failure value that exceeds abandonment threshold, if such solution exists, is replaced with a new randomly created solution. It can be concluded that in our modified scout bee phase the number of scout bees is changeable. Such modified scout bee phase increases the exploration of the search space, but only every $ISRP$ iterations, which helps to leave the local minima.

As in the ABC algorithm, the UABC algorithm does not start with the feasible initial population since initialization with feasible solutions is very hard and in some cases impossible to achieve randomly. Structure of the algorithm already directs the solutions to feasible region due to the Deb's rules.

The pseudo-code of the UABC algorithm is:

```
1.  Initialize the population solutions $x_{ij}$, $i = 1 \ldots, SP/2$, $j = 1 \ldots, D$:
        $x_{ij} = l_j + \delta * (u_j - l_j)$                                        (7)
    where $j \in 1, 2, \ldots, D$, $l_j$ and $u_j$ are the lower and upper bound of the parameter $x_{ij}$ and $\delta$ is a random number in the range [0, 1)
    Initialize $failure_i = 0$, $i = 1 \ldots, SP/2$
2.  Evaluate fitness value of the population by Eq. (5)
3.  $cycle$ =0
4.  **repeat**
5.  Produce a new solution $v_i$ for each employed bee:
        $\begin{cases} x_{ij} = x_{ij} + \varphi_i * (x_{ij} - x_{kj}), & \text{if } R_j < MRE \\ x_{ij}, & \text{otherwise} \end{cases}$     (8)
    where $\varphi_i$ is random number in range [−1,1),
    $k \in \{1, 2, \ldots, SP/2\}$, $k \neq i$ is randomly chosen index,
    $R_j$ is randomly chosen real number in range [0,1) and $j = 1, 2, \ldots, D$ ($D$ is dimension of the problem)
6.  Apply selection process based on Deb's method
7.  If solution $x_i$ does not improve $failure_i = failure_i + 1$, otherwise $failure_i = 0$
8.  Calculate the probability values $P_i$ for the solutions $x_{ij}$:
        $P_i = 0.9 * (fitness_i / maxfit) + 0.1$                                  (9)
    where *maxfit* is the best fitness value of the population. Values 0.9 and 0.1 in Eq. (9) ensure some diversity in the case when one solution is superior to all others by allowing 10% of onlookers to select different solution.
9.  Produce the new solutions $v_i$ for the onlookers from the solutions $x_{ij}$ selected depending on $P_i$:
        $\begin{cases} x_{ij} = x_{ij} + \varphi_i * (x_{ij} - x_{kj}), & \text{if } R_j < MRO \\ x_{ij}, & \text{otherwise} \end{cases}$     (10)
    and evaluate their fitness value
10. Apply selection process based on Deb's method
11. If solution $x_i$ does not improve $failure_i = failure_i + 1$, otherwise $failure_i = 0$
12. **If** (*cycle mod ISRP* = 0) every infeasible solution and one feasible solution with the highest failure value that exceeds abandonment threshold (if such solutions exist) are replaced, each with a new randomly produced solution $x_{ij}$ by Eq. (7)
    **else**
    only one solution (infeasible or feasible) with the highest failure value that exceeds abandonment threshold, if such solution exists, is replaced with a new randomly created solution
13. Memorize the best solution achieved so far
14. $cycle$ = $cycle$+ 1
15. **until** $cycle = MCN$
```

The state variables were treated in the UABC algorithm as follows: for continuous variables, initial values were generated randomly between upper and lower bounds of the specification values. The value was also modified in the employed and onlooker bee's phases between the bounds. For discrete variables, they could be handled in Equations (7), (8) and (10) with a small modification, i.e., as though they were continuous with nearest available discrete values then being chosen. In that way, both continuous and discrete numbers can be handled by the algorithm with no inconsistency.

The key difference between UABC and ABC algorithm, as explained, is in different balance of exploitation and exploration, on the global level as onlooker bees and scouts, as well within these phases. The exploitation in the onlooker phase is improved by limiting new solutions only to the diagonal between two solutions and by decreasing MRO parameter that limits the number of parameters that will be changed, while at the same time diversity is added by different selection probability formula that relaxes suppression of infeasible solutions. The exploration at the scout phase is increased by allowing more scouts periodically that helps to leave the local minima.

Benchmark problems

In order to evaluate the performance of the UABC algorithm we used five well-known standard engineering problems: welded beam, pressure vessel, tension/compression spring, speed reducer and gear train, same as used by Akay and Karaboga (2010). Additionally, since Akay and Karaboga (2010) compare to SCA (Ray and Liew 2003) which also includes the three-bar truss problem, we included that problem. Akay and Karaboga (2010) do not compare the welded beam problem with SCA (Ray and Liew 2003) and PSO (He et al. 2004) algorithms since they used different version of

Table 1 Summary of main properties of the benchmark problems

Problem	D	LI	NI
Welded beam	4	2	5
Pressure vessel	4	3	1
Tension/compression spring	3	1	3
Speed reducer	7	4	7
Gear train	4	0	0
Three-bar truss	2	0	3

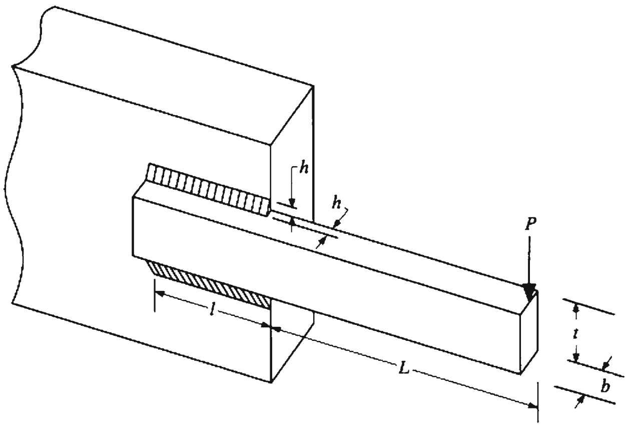

Fig. 1 Welded beam design structure

that problem. We also included that second version of the welded beam problem.

These non-linear engineering design problems have discrete and continuous variables. Discrete variables are used in many ways such as the representation of the set of standard sized components, the decision on the number of identical parts or the choice between different design options. The number of optimization parameters (D), the number of linear inequalities (LI) and nonlinear inequalities (NI) are given in Table 1.

The welded beam design problem consists in dimensioning a welded steel beam and the welding length so as to minimize its cost subject to constraints on shear stress, τ, bending stress in the beam, σ, buckling load on the bar, Pc, end deflection of the beam, δ, and side constraints. The beam has a length of 14 in and 6000 lb force is applied at the end of the beam. There are four continuous variables: x_1, x_2, x_3, x_4 which in structural engineering are commonly symbolized by the letters shown in Fig. 1.

The design variables are thickness of the weld h, length of the weld l, width of the beam t, and thickness of the beam b. The welded beam problem consists of a nonlinear objective function, five nonlinear and two linear inequality constraints. The solution is located on the boundaries of the feasible region. The ratio of feasible region to entire search space is quite small for welded beam problem. Since the problem has four continuous variables it is a continuous constrained optimization problem.

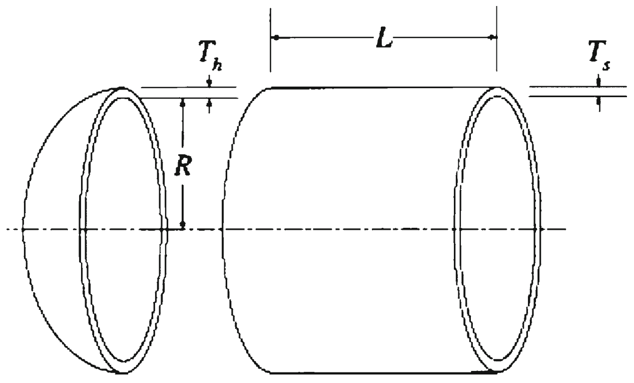

Fig. 2 Pressure vessel design

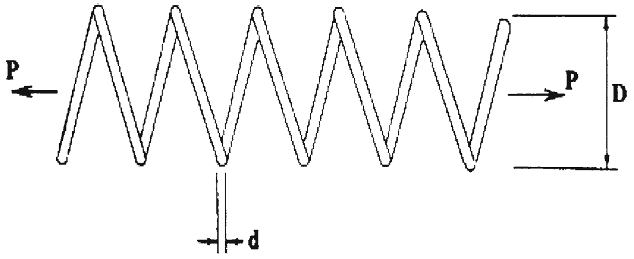

Fig. 3 Tension/compression spring

The pressure vessel problem is to design a compressed air storage tank with a working pressure of 3000 psi and a minimum volume of 750 ft^3. The schematic of a pressure vessel is shown in Fig. 2.

A cylindrical vessel is capped at both ends by hemispherical heads. Using rolled steel plate, the shell is made in two halves that are joined by two longitudinal welds to form a cylinder. Each head is forged and then welded to the shell. Let the design variables be denoted by the vector $x = [x_1, x_2, x_3, x_4]^T$, where x_1 is the spherical head thickness, x_2 is the shell thickness, x_3 and x_4 are the radius and length of the shell, respectively. The objective is to minimize the manufacturing cost of the pressure vessel. The manufacturing cost of pressure vessel is a combination of material cost, welding cost and forming cost. The design variables x_1 and x_2 have to be integer multiples of 0.0625 inch which are the available thickness of rolled steel plates. The radius x_3 and the length x_4 are continuous variables. Pressure vessel problem has a nonlinear objective function, a nonlinear and three linear inequality constraints. Since the problem has two discrete variables and two continuous variables it is a mixed discrete-continuous constrained optimization problem.

The tension/compression spring problem is shown in Fig. 3.

It minimizes the weight of a tension/compression spring, subject to constraints of minimum deflection, shear stress,

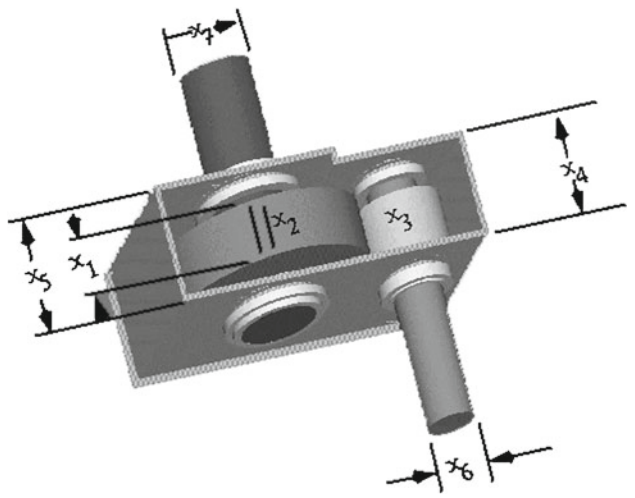

Fig. 4 Speed reducer

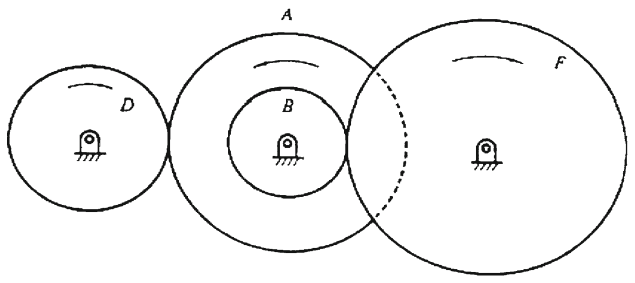

Fig. 5 Gear train design

surge frequency, and limits on outside diameter and on design variables. There are three continuous variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. This problem has a nonlinear objective function, a linear and three nonlinear inequality constraints.

The speed reducer problem represents the design of a simple gear box such as might be used in a light airplane between the engine and propeller to allow each to rotate at its most efficient speed. The design of the speed reducer shown in Fig. 4, is considered with the face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7.

Speed reducer problem has seven nonlinear and four linear constraints. Four constraints are active at the best known feasible solution. Since the problem has one discrete variable and six continuous variables it is a mixed discrete-continuous constrained optimization problem. The best known feasible solution [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] producing a 2994.34 kg gearbox.

The gear train problem is to optimize the gear ratio for the compound gear train arrangement shown in Fig. 5.

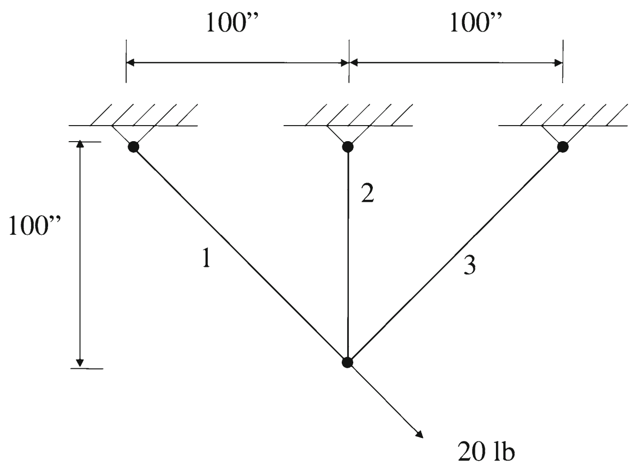

Fig. 6 Three-bar truss design

The gear ratio for a reduction gear train is defined as the ratio of the angular velocity of the output shaft to that of the input shaft. In order to produce the desired overall gear ratio, the compound gear train is constructed out of two pairs of the gearwheels, $d-a$ and $b-f$. The overall gear ratio between the input and output shafts can be expressed as:

$$i_{tot} = \frac{w_0}{w_i} = \frac{z_d z_b}{z_a z_f} \tag{11}$$

where w_0 and w_i are the angular velocities of the output and input shafts, respectively, and z denotes the number of teeth on each gearwheel. It is desirable to produce a gear ratio as close as possible to 1/6.931. For each gear, the number of teeth must be from 12 to 60. Variables to be optimized are in discrete form since each gear has to have an integral number of teeth.

Three-bar truss problem shown in Fig. 6 can be stated as a single objective optimization problem with minimization of its weight, subject to stress constraints in the three members.

The truss is subjected to a concentrated load, and the cross-sectional areas of members 1 and 3 are the same. Two design variables, x_1 and x_2 are the cross-sectional areas of members 1 and 2, respectively. Since the problem has two continuous variables and three nonlinear inequality constraints it is a continuous constrained optimization problem. The best known feasible solution is $f(0.788675, 0.408248) = 263.895843$.

The second version of the welded beam design problem used by some authors differs from the first version in two constraints.

The mathematical models of the benchmark problems are given as appendixes for the first five problems in Akay and Karaboga (2010) and for additional two problems in Ray and Liew (2003).

Table 2 The values of the control parameters of the algorithms

ABC		UABC	
SP	30	*SP*	60
MCN	1,000	*MCN*	250
MR	0.9	*MRE*	0.9
		MRO	0.5
SPP	400	*ISRP*	100
limit	*SP***D**5	*limit*	*SP***D**0.5

Experimental study

The proposed UABC algorithm has been implemented in Java programming language. We used JDK (Java Development Kit) version 6 and Eclipse platform SDK 3.4.2 to develop the application which includes user-friendly graphical user interface (GUI). Tests for seven engineering benchmark problems were done on an Intel(R) Core(TM) 2 Duo CPU E8500@4-GHz computer with 4 GB RAM memory.

The values of the algorithm-specific control parameters are given in Table 2 where parameters are: SP—size of bee population, MCN—maximum cycle number, MR—modification rate, MRE—modification rate for employees, MRO—modification rate for onlookers, SPP—scout production period, $ISRP$—infeasible solution replacement period, limit—limit for nonimprovement and D—dimension of the problem.

We compare our results to the state-of-the-art ABC algorithm for engineering constrained problems (Akay and Karaboga 2010) where their results were favorably compared to SCA (Ray and Liew 2003), PSO (He et al. 2004), $(\mu + \lambda) - ES$ (Mezura-Montes and Coello Coello 2005) and UPSOm (Parsopoulos and Vrahatis 2005) algorithms. The results of the experiments for the UABC algorithm, as well as comparative results for the ABC algorithm from Akay and Karaboga (2010) are presented in Table 3.

For each problem we performed 30 independent runs, as in Akay and Karaboga (2010) but for our algorithm a different ratio of SP and MCN was more suitable. The best results show the ability of an algorithm to find the optimal result, while mean and standard deviation values determine the robustness of the algorithm. The maximum number of evaluations can be considered as a convergence rate.

From Table 3 it can be seen that the UABC algorithm reached better or equal best results than ABC algorithm for all five engineering design problems (best results are bold) with the lower number of evaluations for the first four problems. Both algorithms are of the same computational complexity but the UABC reduced the number of fitness function evaluations by a factor of 2, from 30,000 to 15,000. For the fifth problem, gear train design, in Akay and Karaboga (2010) we believe that there is an error in the table since 60 evaluations are reported in the table, while in the text 60 generations are mentioned. We believe that 60 fitness function evaluations is too few considering that the problem has around 6 million possible solutions (or around 1,5 million if symmetry is considered). For the reported colony size of 30, algorithm would have to find the optimum solution in just 2 iterations which is impossible considering that algorithm starts with random points.

For the mean results and standard deviations, the UABC algorithm showed better performance than the ABC for all five benchmarks, therefore UABC is more robust than the ABC algorithm. UABC algorithm has faster convergence for all tested problems (unknown for the gear train problem, as explained before).

The results for SCA, PSO, $(\mu + \lambda)$-ES and UPSOm are from Akay and Karaboga (2010) and even though ABC compared favorably to them, it was not uniformly better. The UABC algorithm, however, is better or equal compared to the mentioned algorithms also in the cases where ABC was not. For speed reducer problem the UABC algorithm achieved the best known result. For the pressure vessel problem, the evolution strategy $(\mu + \lambda)$-ES (Mezura-Montes and Coello Coello 2005) reports better result in the table, but in the same paper in another table where solution vector is reported the value for the solution function is the same that we achieved and which is generally known to be the best result.

As mentioned in the benchmark description section, since Akay and Karaboga (2010) compare to SCA (Ray and Liew 2003) which also includes the three-bar truss problem, we included that problem also and obtained better results. Also, Akay and Karaboga (2010) do not compare welded beam problem with SCA (Ray and Liew 2003) and PSO (He et al. 2004) algorithms since they used different version of that problem. We included that second version of the welded beam problem and also obtained better results than mentioned algorithms.

Tables 4, 5, 6, 7 and 8 show the solution vectors of the best solution reached by ABC and UABC algorithms and the values of the constraints for each of the problems tested (in Table 9 comparison is to SCA algorithm). Objective function best value was equal for welded beam, tension/compression spring and gear train problems, with slight improvement for pressure vessel and three-bar truss problems and more significant improvement for speed reducer problem for which, as mentioned before, we obtained the best known result.

Conclusion

Artificial bee colony optimization has been proven to be an effective method for solving many hard optimization problems. In this article we presented an upgraded artificial bee

Table 3 Statistical results of the SCA (Ray and Liew 2003), PSO (He et al. 2004), $(\mu + \lambda)$-ES (Mezura-Montes and Coello Coello 2005), UPSOm (Parsopoulos and Vrahatis 2005), ABC (Akay and Karaboga 2010) and UABC algorithms for seven standard engineering problems (best results bold)

Prob.	Stats	SCA	PSO	$(\mu + \lambda)$-ES	UPSOm	ABC	UABC
Welded	Best	NA	NA	**1.724852**	1.92199	**1.724852**	**1.724852**
beam	Mean	NA	NA	1.777692	2.83721	1.741913	**1.724853**
	SD	NA	NA	8.8E−02	0.682980	3.1E−02	**1.7E−06**
	Eval.	NA	NA	30,000	100,000	30,000	**15,000**
Pressure	Best	6171.00	6059.7143	**6059.701610**	6544.27	6059.714736	6059.714335[a]
vessel	Mean	6335.05	6289.92881	6379.938037	9032.55	6245.308144	**6192.116211**
	SD	NA	3.1E+02	2.1E+02	995.573	2.05E+02	**2.04E+02**
	Eval.	20,000	30,000	30,000	100,000	30,000	**15,000**
Tension/	Best	0.012669	**0.012665**	0.012689	0.0131200	**0.012665**	**0.012665**
compress.	Mean	0.012923	0.012702	0.013165	0.0229478	0.012709	**0.012683**
spring	SD	5.9E−04	4.1E−05	3.9E−04	0.00720571	0.012813	**3.31E−05**
	Eval.	25,167	**15,000**	30,000	100,000	30,000	**15,000**
Speed	Best	2994.744241	NA	2996.348094	NA	2997.058412	**2994.471066**
reducer	Mean	3001.758264	NA	2996.348094	NA	2997.058412	**2994.471072**
	SD	4.0E+0	NA	0	NA	0	**5.98E−06**
	Eval.	54,456	NA	30,000	NA	30,000	**15,000**
Gear	Best	NA	NA	NA	**2.700857E−12**	**2.700857E−12**	**2.700857E−12**
train	Mean	NA	NA	NA	3.80562E−08	3.641339E−10	**3.500171E−10**
	SD	NA	NA	NA	1.09631E−07	5.525811E−10	**4.690420E−10**
	Eval.	NA	NA	NA	100,000	**60**	3,000[1]
Three-	Best	263.895847	NA	NA	NA	NA	**263.895843**
bar	Mean	263.903357	NA	NA	NA	NA	**263.895843**
truss	SD	1.26E−02	NA	NA	NA	NA	**0**
	Eval.	17,610	NA	NA	NA	NA	**12,000**
Welded	Best	2.385435	**2.380957**	NA	NA	NA	**2.380957**
beam	Mean	3.255137	2.381932	NA	NA	NA	**2.380973**
(2. ver)	SD	9.59E−01	5.24E−03	NA	NA	NA	**7.73E−05**
	Eval.	33,095	30,000	NA	NA	NA	**18,000**

[a] Explained in the text

colony algorithm (UABC) for constrained engineering problems. The proposed approach was tested on seven standard engineering benchmark problems. The obtained results were compared with the latest results reached by the ABC algorithm extended to solve engineering design problems (Akay and Karaboga 2010) which has been compared to other optimization algorithms and was considered a promising tool for solving that kind of problems. UABC proved to be more robust and outperformed ABC algorithm in terms of best results, robustness and with faster convergence.

For seven benchmark problems (welded beam, pressure vessel, tension/compression spring, speed reducer, gear train, three-bar truss and the second version of welded beam) and four results and parameters for each of these problems (best solution, mean solution and standard deviation for all runs and number of evaluations of the fitness function) i.e. 28 different experiment results, the proposed UABC algorithm produced 26 results that are better (in few cases equal) compared to all five other algorithms. In two cases where it seems that UABC was not the best, there are some unclear details

Table 4 Parameter and constraint values of the best solutions obtained for welded beam problem

	ABC	UABC
x_1	0.205730	0.205730
x_2	3.470489	3.470489
x_3	9.036624	9.036624
x_4	0.205730	0.205730
$g_1(x)$	0.000000	−0.000028
$g_2(x)$	−0.000002	−0.000025
$g_3(x)$	0.000000	−0.000000
$g_4(x)$	−3.432984	−3.432984
$g_5(x)$	−0.080730	−0.080730
$g_6(x)$	−0.235540	−0.235540
$g_7(x)$	0.000000	−0.000050
$f(x)$	1.724852	1.724852

Table 5 Parameter and constraint values of the best solutions obtained for pressure vessel problem

	ABC	UABC
x_1	0.8125	0.8125
x_2	0.4375	0.4375
x_3	42.098446	42.098446
x_4	176.636596	176.636596
$g_1(x)$	0.000000	−0.000000
$g_2(x)$	−0.035881	−0.035881
$g_3(x)$	−0.000226	−0.000000
$g_4(x)$	−63.363404	−63.363404
$f(x)$	6059.714339	6059.714335

Table 6 Parameter and constraint values of the best solutions obtained for tension/compression spring problem

	ABC	UABC
x_1	0.051749	0.051691
x_2	0.358179	0.356769
x_3	11.203763	11.285988
$g_1(x)$	−0.000000	−0.000000
$g_2(x)$	−0.000000	−0.000000
$g_3(x)$	−4.056663	−4.053886
$g_4(x)$	−0.726713	−0.727694
$f(x)$	0.012665	0.012665

Table 7 Parameter and constraint values of the best solutions obtained for speed reducer problem

	ABC	UABC
x_1	3.499999	3.500000
x_2	0.7	0.7
x_3	17	17
x_4	7.3	7.3
x_5	7.8	7.715320
x_6	3.350215	3.350215
x_7	5.287800	5.286654
$g_1(x)$	−0.073915	−0.073915
$g_2(x)$	−0.197999	−0.197999
$g_3(x)$	−0.499172	−0.499172
$g_4(x)$	−0.901555	−0.904644
$g_5(x)$	0.000000	−0.000000
$g_6(x)$	0.000000	−0.000000
$g_7(x)$	−0.7025	−0.7025
$g_8(x)$	0.000000	−0.000000
$g_9(x)$	−0.583333	−0.583333
$g_{10}(x)$	−0.051326	−0.051326
$g_{11}(x)$	−0.010695	−0.000000
$f(x)$	2997.058412	2994.471066

Table 8 Parameter and constraint values of the best solutions obtained for gear train problem

	ABC	UABC
x_1	49	49
x_2	16	16
x_3	19	19
x_4	43	43
$f(x)$	0	0

Table 9 Parameter and constraint values of the best solutions obtained for three-bar truss problem

	SCA	UABC
x_1	0.788621	0.788675
x_2	0.408401	0.408248
$g_1(x)$	NA	0.000000
$g_2(x)$	NA	−1.464102
$g_3(x)$	NA	−0.535898
$f(x)$	263.895847	263.895843

in the articles with which we compare, so the solutions we reached may be the best.

The UABC algorithm achieved better results since it more precisely controls balance of exploitation and exploration, on the global level as onlooker and scout phase, as well within these phases. The exploitation in the onlooker phase is improved by limiting new solutions only to the diagonal between two solutions and by decreasing MRO parameter that limits the number of parameters that will be changed, while at the same time diversity is added by different selection probability formula that relaxes suppression of infeasible solutions. The exploration at the scout phase is increased by allowing more scouts periodically that helps to leave the local minima.

From this work, it can be concluded that our upgraded artificial bee colony algorithm can be efficiently used for solving engineering design problems due to its simplicity, reliability and robustness. Future research will include additional adjustments of algorithm exploitation/exploration mechanisms, with continued respect to a generally accepted rule that any change should be universal, i.e. not adjusted for any specific problem.

Acknowledgments This research is supported by Ministry of Education and Science of Republic of Serbia, Grant no. III-44006.

References

Akay, B., & Karaboga, D. (2010). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-010-0393-4 (Published online).

Aydin, M. E. (2010), Coordinating metaheuristic agents with swarm intelligence. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-010-0435-y (Published online).

Baykasoglu, A., Ozbakir, L., & Tapkan, P. (2007). Artificial bee colony algorithm and its application to generalized assignment problem. In F. T. S Chan & M. K. Tiwari (Eds.), *Swarm intelligence, focus on ant and particle swarm optimization* (pp. 113–144). Vienna: I-Tech Education and Publishing.

Chang, F. C., & Huang, H. C. (2010). A refactoring method for cache-efficient swarm intelligence algorithms. *Information Sciences* doi:10.1016/j.ins.2010.02.025 (Article in press).

Cheshmehgaz, H. R., Desa, M. I., & Wibowo, A. (2011). A flexible three-level logistic network design considering cost and time criteria with a multi-objective evolutionary algorithm. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-011-0584-7 (Published online).

Deb, K. (2000). An efficient constraint-handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186*(2–4), 311–338.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *Biosystems, 43*(2), 73–81.

Gaitonde, V. N., & Karnik, S. R. (2010). Minimizing burr size in drilling using artificial neural network (ann)-particle swarm optimization (pso) approach. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-010-0481-5 (Published online).

Haddad, O. B., Afshar, A., & Marino, M. A. (2006). Honey-bees mating optimization (hbmo) algorithm: A new heuristic approach for water resources optimization. *Water Resources Management, 20*(5), 661–680.

Hamida, S. B., & Schoenauer, M. (2002). Aschea: New results using adaptive segregational constraint handling. In *Proceedings of the congress on evolutionary computation 2002 (CEC'2002)* (pp. 884–889). IEEE Service Center.

He, S., Prempain, E., & Wu, Q. (2004). An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization, 36*(5), 585–605.

Jovanovic, R., & Tuba, M. (2011). An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. *Applied Soft Computing, 11*(8), 5360–5366.

Jun-Qing Li, Q. K. P, & Xie, S. X. (2010). A hybrid variable neighborhood search algorithm for solving multi-objective flexible job shop problems. *Computer Science and Information Systems, 7*(4), 907–930.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D., & Akay, B. (2011). A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing, 11*(3), 3021–3031.

Karaboga, D., & Basturk, B. (2007a). Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *LNAI 4529: IFSA'07 proceedings of the 12th international fuzzy systems association world congress on foundations of fuzzy logic and soft computing* (pp. 789–798). Springer.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization, 39*(3), 459–471.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing, 8*(1), 687–697.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE international conference on neural networks* (pp. 1942–1948). Piscataway, NJ: IEEE Service Center.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation, 7*(1), 19–44.

Lu, M., & Romanowski, R. (2011). Multi-contextual ant colony optimization of intermediate dynamic job shop problems. *The International Journal of Advanced Manufacturing Technology* 1–15. doi:10.1007/s00170-011-3634-6 (Published online).

Mezura-Montes, E., & Coello Coello, C. A. (2005). Useful infeasible solutions in engineering optimization with evolutionary algorithms. In *MICAI 2005: Advances in artificial intelligence of lecture notes in computer science* (pp. 652–662). Springer.

Mezura-Montes, E., & Miranda-Varela, C. A. (2005). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation, 9*(1), 1–17.

Mezura-Montes, E., Coello Coello, E. M., & Gomez-Ramon, R. (2010). Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences, 180*(22), 4223–4262.

Pan, D., & Liu, Z. (2011). An improved particle swarm optimization algorithm. In H. Deng, D. Miao, F. L. Wang, & J. Lei (Eds.), *Emerging research in artificial intelligence and computational intelligence, communications in computer and information science* (Vol. 237, pp. 550–556). Berlin: Springer.

Parsopoulos, K., & Vrahatis, M. (2005). Unified particle swarm optimization for solving constrained engineering optimization problems. In *ICNC 2005: Advances in natural computation, volume 3612/2005 of LCNS* (pp. 582–591). Springer.

Pasandideh, S. H. R., Niaki, S. T. A., & Hajipour, V. (2011). A multi-objective facility location model with batch arrivals: Two parameter-tuned meta-heuristic algorithms. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-011-0592-7 (Published online).

Pham, D. T., Kog, E., Ghanbarzadeh, A., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm—a novel tool for complex optimisation problems. In *IPROMS 2006 proceeding 2nd international virtual conference on intelligent production machines and systems* (pp. 454–459). Elsevier.

Puranik, P., Bajaj, P., Abraham, A., Palsodkar, P., & Deshmukh, A. (2011). Human perception-based color image segmentation using comprehensive learning particle swarm optimization. *Journal of Information Hiding and Multimedia Signal Processing, 2*(3), 227–235.

Ray, T., & Liew, K. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation, 7*(4), 386–396.

Runarsson, T. P., & Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation, 4*(3), 284–294.

Runarsson, T. P., & Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics Part C-Applications and Reviews, 35*(2), 233–243.

Srinivasan, D., & Seow, T. (2003). Particle swarm inspired evolutionary algorithm (ps-ea) for multiobjective optimization problems. In: *The 2003 congress on evolutionary computation—CEC 2003* (pp. 2292–2297). IEEE Press.

Wang, Y., Zhang, B., & Chen, Y. (2011). Robust airfoil optimization based on improved particle swarm optimization method. *Applied Mathematics and Mechanics (English Edition), 32*(10), 1245–1254.

Yang, X. S. (2005). Engineering optimizations via nature-inspired virtual bee algorithms. In *Artificial intelligence and knowledge engineering applications: A bioinspired approach, LNCS* (Vol. 3562, pp. 317–323). Springer.

Zavala, A. E. M., Hernandez, A., & Diharce, E. R. V. (2005). Constrained optimization via particle evolutionary swarm optimization algorithm (peso). In *GECCO '05 Proceedings of the 2005 conference on genetic and evolutionary computation* (pp. 209–216). ACM Press.