Journal of Engineering Science and Technology
Vol. 10, No. 1 (2015) 111 - 133

IMPROVED SEARCH MECHANISM IN ABC AND ITS APPLICATION IN ENGINEERING DESIGN PROBLEMS

TARUN KUMAR SHARMA[1,*], MILLIE PANT[2]

[1]Amity Institute of Information Technology, Amity University Rajasthan, India
[2]Department of Applied Science and Engineering, IIT Roorkee, India
*Corresponding Author: taruniitr1@gmail.com

Abstract

ABC (Artificial Bee Colony) is one of the most recent nature inspired algorithm (NIA) based on swarming metaphor. Proposed by Karaboga in 2005, ABC has proven to be a robust and efficient algorithm for solving global optimization problems over continuous space. In this paper, we propose a modified version of the ABC. In modified version called Dichotomous ABC (DABC), the idea is to move dichotomously in both directions to generate a new trial point. The results of a trade study carried out on three classical structural optimization problems taken from literature indicate the validity of the proposed algorithm.

Keywords: Artificial bee colony, Dichotomous optimization, Constrained optimization, Engineering design problems.

1. Introduction

In the past few decades several nature inspired algorithms (NIA) have emerged as a potential tool for solving global optimization problems. NIA may be classified as the ones based on natural evolution, commonly known as Evolutionary Algorithms (EA) and the ones that are based on behavioral pattern displayed by various species, particularly the ones that live in groups (or swarms). In the present study, the focus is on ABC, which is one of the recently proposed NIA introduced by Karaboga in 2005 [1-3]. ABC follows the analogy of the socio-cooperative behavior demonstrated by honey bees in their search for nectar. A brief overview of the working of ABC algorithm is given in Section 2.

In this paper, we have embedded a concept of dichotomous search for locating the food sources in ABC. The proposed variant named DABC is applied to constrained

real engineering optimization problems taken from literature and its performance is compared with the performance of various variants taken from literature.

Here we would like to mention that a preliminary version of this work has been presented in a conference [4], where the proposal was just tested on unconstrained benchmark functions. However, in this study we present its elaborated version and implemented it to solve engineering design problems. Here we provide a comprehensive set of experimental verifications of the proposed DABC.

The remaining of the paper is organized as follows. Section 3, introduces the Artificial Bee Colony. Proposed DABC algorithm is discussed in Section 4. The engineering design problems and implementation of the proposed DABC is discussed in Section 5. In Section 6, Parameter settings, algorithms used to compare the results and result discussion is presented. Finally, there are the conclusions of the paper.

2. Survey of ABC Literature

Karaboga and Basturk compared the performance of the ABC algorithm with the performance of other well-known modern heuristic algorithms such as genetic algorithm (GA), differential evolution (DE), particle swarm optimization on unconstrained and constrained problems [3, 5]. Baykasoglu et al. [6] incorporated the ABC algorithm with shift neighbourhood searches and greedy randomized adaptive search heuristic and applied it to the generalized assignment problem. Pan et al. [7] integrated a search iteration operator based on the fixed point theorem of contractive mapping in Banach spaces with the ABC algorithm in order to improve convergence rate. ABC, however, like its counterpart population based search algorithms suffers from some inherent drawbacks like slow or premature convergence while dealing with certain complex models [8]. In order to maximize the exploitation capacity of the onlooker stage [9] introduced the Newtonian law of universal gravitation in the onlooker phase of the basic ABC algorithm in which onlookers are selected based on a roulette wheel.

Haijun and Qingxian [10] proposed a modification in the initialization scheme by making the initial group symmetrical, and the Boltzmann selection mechanism was employed instead of roulette wheel selection for improving the convergence ability of the ABC algorithm. Kang et al. [11] proposed a hybrid simplex artificial bee colony algorithm which combines Nelder-Mead simplex method to solve inverse analysis problems. Alatas [12] used different chaotic maps for parameter adaptation in order to improve the convergence characteristics and to prevent the ABC to get stuck on local solutions. In order to further improve the performance of ABC, several modifications are available in literature. Zhu and Kwong [13] proposed an improved ABC algorithm called gbest-guided ABC by incorporating the information of global best solution into the solution search equation. Lei et al. [14] proposed some modification on the original iteration equation of ABC, in which an inertial weight is added on the first item to balance the local and the global search, inspired by the improved strategies of PSO and applied the improved ABC to data clustering problem.

Banharnsakun et al. [15] introduced the best-so-far selection in ABC and assessed its performance on two sets of problems: numerical benchmark functions

and image registration applications. Tuba et al. [16] presented a novel algorithm named GABC which integrates ABC algorithm with self-adaptive guidance adjusted for engineering optimization problems. Li et al. [17] proposed an improved ABC algorithm in which inertia weight and acceleration coefficients are introduced to improve the search process of ABC algorithm. An improved ABC algorithm by the inspiration of DE algorithm was proposed by [18] and the author demonstrated its good performance in solving complex numerical optimization problems. ABC is also applied to solve time tabling [19] and course scheduling problem [20]. Various variants to improve the convergence rate of ABC are proposed [21, 22].

While considering the application part of ABC, the ABC algorithm also was extended for constrained optimization problems in [5] and was applied to train neural networks [23-25]. Rao et al. [26] applied the ABC algorithm to network reconfiguration problem in a radial distribution system in order to minimize the real power loss, improve voltage profile and balance feeder load subject to the radial network structure in which all loads must be energized. Singh [27] in 2009 used the ABC algorithm for the leaf-constrained minimum spanning tree problem and compared the approach against genetic algorithm (GA), ant colony optimization (ACO) and tabu search (TS) and to solve TSP problems [28]. [29-30] applied ABC to medical pattern classification and clustering problems. A comprehensive survey on ABC is discussed by Karaboga in [31]. Pan et al. [7] in 2012 proposed a discrete ABC algorithm in combinational optimization for flow shop scheduling problem. ABC with its increasing popularity is widely applied to solve various industrial problems [21], [32-35].

3. Introduction to ABC

ABC algorithm, inspired by the intelligent food foraging behavior of honey bees, is one of the recent swarm intelligence algorithm explored in the literature. In ABC algorithm the position of the food source denotes the solution of the problem. The quality of the food source signifies the fitness of the solution. The colony of honey bees in ABC algorithm are categorized into three groups namely employed bees, onlooker bees and scout bees. First half of the colony consists of the employed bees and the second half includes the onlookers. Each employed bee is associated with a food source i.e. the number of employed bees are equal to the number of food sources. Once the food source of the employed bee abandons it become the scout bee.

In ABC system, artificial bees fly around in the search space, and some (employed and onlooker bees) choose food sources depending on the experience of themselves and their nest mates and adjust their positions. Some (scouts) fly and choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one. Thus, ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, attempting to balance exploration and exploitation process.

The solution searching process in ABC is an iterative process, similar to other nature inspired or metaheuristic algorithms. This process is elaborated step wise in Section 3.1.

Initially, ABC was developed for solving unconstrained optimization problems. However, with the help of some simple changes it can be easily modified for solving constrained problems as well, which is described in Section 3.2.

3.1. Unconstrained ABC

Step-by-step procedure of ABC is given as follows:

A) Randomly distributed food sources are generated over a *D*-dimensional search space.

B) An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, P_i, calculated by the expression:

$$P_i = \frac{fit_i}{\sum_{n=1}^{NP} fit_n} \tag{1}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and NP is the number of food sources which is equal to the number of employed bees. In order to produce a candidate food position from the old one in memory, the ABC uses the following expression:

$$v_{ij} = x_{ij} + r_{ij}(x_{ij} - x_{kj}) \tag{2}$$

where v_{ij}, G in the neighborhood of x_{ij}, G, are new solutions (food source positions), $k \in \{1, 2, \ldots, NP\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indexes. Further, $k \neq i$ and r_{ij} is a random number between [-1, 1].

C) In ABC algorithm, "*limit*" is an important control parameter, it controls the times of updates of a certain solution. If a solution cannot be improved further through a predetermined number of cycles called limit then that solution is assumed to be abandoned, and the employed bee becomes a scout. If the solution is x_i and $j \in \{1, 2, \ldots, D\}$to be abandoned, then a new solution produced randomly would replace x_i by:

$$x_{ij} = x_{\min}^{j} + rand(0,1)(x_{\max}^{j} - x_{\min}^{j}) \tag{3}$$

where $x_{\min}^{j}$ is the lower bound of the parameter j and $x_{\max}^{j}$ is the upper bound of the parameter j.

3.2. Constrained ABC

In order to adapt ABC algorithm for solving constraint optimization problems, the constraint handling methods proposed by Deb [36] are adopted instead of greedy selection process in unconstrained ABC algorithm. Deb's method dealing with constraints consists of three simple heuristic rules:

A) Any feasible food position is preferred to any infeasible food position,

B) Among two feasible food positions, the one having better fitness value is preferred,

C) Among two infeasible food positions, the one having smaller constraint violation is preferred.

For constrained optimization, in order to produce a candidate food position (by an employed or an onlooker bee) the following is used [3]:

$$v_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), & if \ R_j \le MR \\ x_{ij} \ , & otherwise \end{cases} \quad (4)$$

where $k \in \{1, 2,..., SN\}$ is randomly chosen index such that k is different from i. R_j – randomly chosen real number in the range [0,1] and $j \in \{1, 2,...,D\}$. MR – modification rate, that controls whether the parameter x_{ij} will be modified or not.

Detailed algorithm is discussed in Fig. 1.

Initialize p_w, the percentage of employed bees
Initialize $n_e^{\max}$ maximum number of explorer bees
Initialize λ, the foraging *limit*
Set n_s as the size of swarm
Set the fittest bee, β to null
Create and initialize to random positions $n_w = n_s p_w$ employed bees
Set the number of onlooker bees, $n_o = n_s - n_w$
for *each worker bee*, w_i **do**
    Set $a_i = 0$, where a_i is the number of failed position updates
end
while *stopping condition is false* **do**
    Set $n_e = 0$, where n_e is the number of explorer bees
    for $i = 1,\ldots,n_w$ **do**
        Randomly select $j \in [1, n_w], j \neq i$
        Create a new position v_{wiwj} from worker bees using Equation (4)
        if $f(v_{wiwj}) < f(x_{wi})$ based on Pareto Ranking Method **then**
            $x_{wi} = v_{wiwj}$
        end
        else
            a_i++
        end
        if $a_i > \lambda$ and $n_e < n_e^{\max}$ **then**
            n_e++
            Move w_i to a new random position in the search space
            $a_i = 0$
        end
    end
    for $i = 1,\ldots, n_o$ **do**
        Select $j \in [1, n_w]$ proportionate to $f(x_{wi})$
        Set $o_i = w_j$
        Select $k \in [1,...,n_w], k \neq j$

Create a new position v_{oiwj} from onlooker bee o_i and worker bee w_j using Equation (4)
if $f(v_{oiwj}) < f(x_{oi})$ based on Pareto Ranking Method **then**

$x_{oi} = v_{oiwj}$
end
end
for *each bee,* b_i*, in the swarm* **do**
if $f(x_{bi} < f(x_\beta)$ then
$\beta = b_i$
end
end
end
Return x_β as Solution.

Fig. 1. Algorithm: Constrained Artificial Bee Colony.

ABC algorithm is used in finding the best food source (solution) from all feasible food sources (solutions). However, ABC can sometimes be slow to converge. In order to improve the algorithm performance, we present a modified search method, which is described in the next section of this paper.

4. Proposed Improved Search Mechanism (Dichotomous-ABC)

The search mechanism in modified ABC traverse in forward as well as in a backward direction in search of optimal food locations, hence it named as Dichotomous ABC (DABC). This is to say that we are using a concept of bidirectional movement (BM) [37] for finding out the optimum solution. The pseudo code of the proposed plan of movement may be given as:

$$\begin{aligned} &if \quad [f(x+d) < f(x)] \rightarrow x = x + d \\ &elseif \quad [f(x-d) < f(x)] \rightarrow x = x - d \\ &else \rightarrow do\ not\ change\ \ x \end{aligned} \tag{5}$$

It can be seen that the proposed method searches both in forward as well as in reverse direction. This method expresses that if movement in the forward direction does not improve the fitness value, with a high probability, we reverse the movement with the hope of getting an improved solution.

In the proposed DABC, BM is used while producing new food source positions. In the new stage, if forward movement, in ABC algorithm, does not improve value of fitness/cost function i.e. $f(v_i) > f(x_i)$, the reverse movement, will be checked as:

$$v'_{ij} = x_{ij} - \Phi_{ij}(x_{ij} - x_{kj}) \tag{6}$$

The modification in the basic ABC algorithm is done by producing a new candidate food position from the old one in memory; using the following expression:

If (rand (0, 1) $\geqslant$ 0.5) *then*

$$v_{ij} = \begin{cases} x_{ij} + r_{ij}(x_{ij} - x_{kj}), & if\ R_{ij} < MR, \\ x_{ij}, & otherwise, \end{cases}$$

Else (7)

$$v_{ij} = \begin{cases} v'_{ij}, & if\ R_{ij} < MR, \\ x_{ij}, & otherwise, \end{cases}$$

End If

On other words, the DABC presents an inversion of the search direction by changing the sign.

5. Implementation of Dichotomous-ABC

Most design optimization problems in structural engineering are highly non-linear, include many different design variables and complicated constraints on stresses, displacements, load carrying capability, and geometrical configuration. Since non-linearity often results in multiple local optima only global algorithms should be used [38]. Another complication is that not all design variables are continuous, and some variables can only take certain discrete values. Mixed continuous/discrete optimization problems usually require search techniques that are problem-specific. In this study the proposed approach, DABC, is applied to three design problems: welded beam design optimization problem, pressure vessel design optimization problem, and helical compression spring design. These nonlinear engineering design problems have discrete and continuous variables.

5.1. Welded beam design

The 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 [39-40]. 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 Figure 2 (h,l,t,b). 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 objective function of the problem is expressed as follows:

$$Minimize:\ f(h,L,t,b) = (1+c_1)h^2 l + c_2 tb(L+l) \tag{8}$$

subject to constraints:

- shear stress (τ)

$$g_1(x) = \tau_d - \tau(x) \geq 0 \tag{9}$$

- bending stress in the beam (σ)

$$g_2(x) = \sigma_d - \sigma(x) \geq 0 \tag{10}$$

- geometric constraints

$$g_3(x) = b - h \geq 0 \tag{11}$$

- buckling load on the bar (P_c)

$$g_4(x) = P_c(x) - P \geq 0 \tag{12}$$

- deflection of the beam (δ)

$$g_5(x) = 0.25 - \delta(x) \geq 0 \tag{13}$$

where

$$\tau(x) = \sqrt{(\tau'(x))^2 + (\tau''(x))^2 + l\tau'(x)\tau''(x)/\sqrt{0.25(l^2 + (h+t)^2)}} \tag{14}$$

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

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

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

$$\tau' = \frac{6000}{\sqrt{2}hl} \tag{18}$$

$$\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\{0.707hl(l^2/12 + 0.25(h+t)^2\}} \tag{19}$$

The simple bounds of the problem are: 0.125≤h≤6 5, 0.1≤*l*, t≤10 and 0.1≤b≤5. The values of parameters involved in the formulation of the welded beam problem are also shown in Table 1.

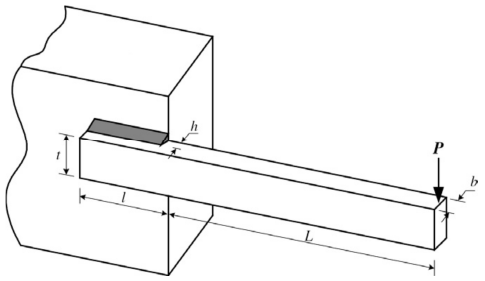

Fig. 2. Schematic of the welded beam design problem.

Table 1. Values of parameters involved in the formulation of the welded beam problem.

Constant item	Description	Values
C_1	Cost per volume of the welded material	0.10471($/in^3)
C_2	Cost per volume of the bar stock	0.04811($/in^3)
τ_d	Design shear stress of the welded material	13600(psi)
σ_d	Design normal stress of the bar material	30000(psi)
δ_d	Design bar end deflection	0.25(in)
E	Young's modulus of bar stock	30 x 10^6 (psi)
G	Shear modulus of bar stock	12 x 10^6 (psi)
P	Loading condition	6000 (lb)
L	Overhang length of the beam	14 (in)

5.2. Pressure vessel design

The cylindrical pressure vessel capped at both ends by hemispherical heads (Figure 3) must be designed for minimum cost. This optimization problem was originally formulated by Sandgren [41]. The compressed air tank has a working pressure of 3000 psi and a minimum volume of 750 ft^3, and must be designed according to the ASME code on boilers and pressure vessels. The total cost results from a combination of welding, material and forming costs. The thickness of the cylindrical skin (T_s), the thickness of the spherical head (T_h), the inner radius (R), and the length of the cylindrical segment of the vessel (L) were included as optimization variables. Thicknesses can only take discrete values which are integer multiples of 0.0625 in. The objective optimization problem can be formulated as follows:

$$Minimize: f(T_s, T_h, R, L) = 0.6224T_s RL + 1.7781T_h R^2 + 3.1661T_s^2 L + 19.84T_h^2 R \quad (20)$$

Constraints are set in accordance with the ASME design codes; g_3 represents the constraint on the minimum volume of 750 ft^3. The constraints are stated as follows:

$$g_1 = -T_s + 0.0193R \le 0 \quad (21)$$

$$g_2 = -T_h + 0.00954R \le 0 \quad (22)$$

$$g_3 = -\pi R^2 L - \frac{4}{3}\pi R^3 + 750 \times 1728 \le 0 \quad (23)$$

$$g_4 = L - 240 \le 0 \quad (24)$$

where 1 x 0.0625≤T_s, T_h≤99 x 0.0625, 10≤R≤200 and 10≤L≤200.

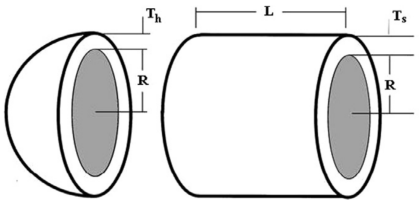

Fig. 3. Schematic of the pressure vessel design problem.

5.3. Helical compression spring design

The cylindrical pressure vessel capped at both ends by hemispherical heads The helical compression spring shown in Figure 4 is subject to an axially guided constant compression load. The spring must be designed for minimum volume. Spring ends are ground and squared. The winding coil diameter (D), wire diameter (d), and the number of spring coils (n) were included as optimization variables: D is continuous, n is an integer, and d can take one of the 42 discrete values listed in Table 3. The cost function to be minimized is the spring volume, expressed as:

$$Minimize: f(D, d, n) = \frac{\pi D d^2 (n+2)}{4} \quad (25)$$

The following eight constraints specify the design limitations:

The design shear stress caused by the compression load should be lower than the allowable maximum shear stress (S) of the material.

$$g_1 = \frac{8C_f P_{\max} D}{3.14156 d^3} - S \le 0 \tag{26}$$

The free length of the spring should be shorter than the maximum specified value L_{free}

$$g_2 = \frac{8KP_{\max} Dn}{C_f d^4} + 1.05(n+2)d - L_{free} \le 0 \tag{27}$$

The wire diameter must not be less than the specified minimum diameter d_{min}

$$g_3 = d_{\min} - d \le 0 \tag{28}$$

The outer diameter of the coil should be smaller than the specified maximum diameter D_{max}

$$g_4 = (d+D) - D_{\max} \le 0 \tag{29}$$

The inner coil diameter must be at least three times less than the wire diameter to avoid a lightly wound spring

$$g_5 = 3 - \frac{D-d}{d} \le 0 \tag{30}$$

The deflection under the given load d must be less than the specified maximum deflection under preload δ_{pm}

$$g_6 = \delta - \delta_{pm} \le 0 \tag{31}$$

The combined deflection must be consistent with the coil free length L_{free}

$$g_7 = \frac{8KP_{\max} D^3 n}{C_f d^4} + \frac{P_{\max} - P_{load}}{K} + 1.05(n+2)d - L_{free} \le 0 \tag{32}$$

The deflection from preload to maximum load must be greater than the specified working deflection δ_W

$$g_8 = \delta_W - \frac{P_{\max} - P_{load}}{K} \le 0 \tag{33}$$

where

$$C_f = \frac{4(S_i)-1}{4(S_i)-4} + \frac{0.615}{S_i} \tag{34}$$

$$S_i = \frac{D}{d} \tag{35}$$

$$K = \frac{Gd^4}{8nD^3} \tag{36}$$

$$\delta_p = \frac{F_p}{K} \tag{37}$$

The values assigned to constant terms involved in the problem statement are listed in Table 2.

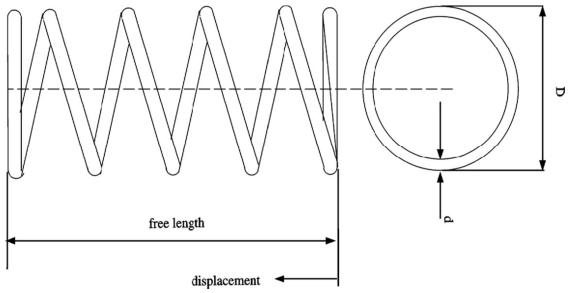

Fig. 4. Schematic of the helical compression spring design problem.

Table 2. Values of parameters involved in the formulation of the helical spring problem.

Constant item	Description	Values
P_{max}	Maximum work load	1000.0(lb)
S	Maximum shear stress	189 x 10^3 (psi)
E	Elastic modulus of the material	30 x 10^6 (psi)
G	Shear modulus of the material	11.5 x 10^6 (psi)
L_{free}	Maximum coil free length	14 (in)
d_{min}	Minimum wire diameter	0.2 (in)
D_{max}	Maximum outside diameter of the spring	3.0 (in)
P_{load}	Preload compression force	300.0 (lb)
δ_{pm}	Maximum deflection under preload	6.0 (in)
δ_W	Deflection from preload position to maximum load position	1.25 (in)

6. Experimental Results and Discussions

6.1. Parameter settings

ABC algorithm has fewer number of control parameters in comparison to other metaheuristic algorithms, these are: colony (swarm) size, limit, number of employed bees, number of onlookers, number of scouts and maximum number of cycles (MCN). The percentages of onlooker bees and employed bees are 50% of the colony size. Each experiment was repeated 30 runs.

Control Parameter	Value
Colony Size (SN)	50
Employed bees	25 (50% of SN)
Onlooker bees	25 (50% of SN)
Scout Bee	one
Limit	SN(2D+1)
Maximum Cycle Numbers (MCN)	4000
Function Evaluation	50000
Modification Rate (MR)	0.4

Random numbers are generated using inbuilt function *rand()* in DEVC++. And the algorithm is executed on Pentium IV, using DEV C++. The performance of DABC is analysed in terms of best, worst mean fitness function values, standard deviation (*SD*) and average generations for all the three real life problems taken.

6.2. Algorithm used for making comparisons

In this study we have compared the performance of proposed DABC with some basic as well as modified or advanced versions of GA, SA, PSO, DE, ABC, FA and a like. Their key characteristics are described as:

Algorithm [Reference]	**Algorithm and Problem Solved**
Branch and Bound [41]	In this paper Helical compression spring design problem was solved using branch and bound procedure in conjunction with either an exterior penalty function or a quadratic programming method.
GA [42]	This paper presents revised genetic operators and a new recombination scheme as well as it also simultaneously introduces an additional selection pressure to increase the speed of convergence. This proposed algorithm is applied to solve Welded beam problem.
GA [43]	This algorithm uses concept of non-dominance (commonly used in multiobjective optimization) as a way to incorporate constraints into the fitness function of a genetic algorithm. Each individual is assigned a rank based on its degree of dominance over the rest of the population. Further it uses a self-adaptation mechanism that avoids the traditional empirical adjustment of the main genetic operators (i.e., crossover and mutation). Welded beam problem is solved as an application of this study.
GA [44]	In this study GA with penalty method constraint handling is used and applied to Welded beam problem.
GA [45]	A parameter-less adaptive penalty scheme for genetic algorithms applied to constrained optimization problems is proposed and applied to solve Welded beam problem. It uses feedback from the evolutionary process the procedure automatically defines a penalty parameter for each constraint.
AIS^1-GA [46]	A genetic algorithm (GA) is hybridized with an artificial immune system (AIS) and applied to Welded beam problem. The AIS is inspired in the clonal selection principle and is embedded into a standard GA search engine in order to help move the population into the feasible region.
SA^2 [47]	In this work simulated annealing (SA), a stochastic global optimization technique, is implemented by augmenting it with

	a feasibility improvement scheme (FIS). FIS is also found to help recover from the infeasible design space rapidly and the results are justified by implementing it on Welded beam problem as an application.
SA-DS^3 [48]	A simulated-annealing-based method called Filter Simulated Annealing (FSA) method is proposed to deal with the constrained global optimization problem, further it is applied to solve Welded beam problem. The FSA method invokes a multi-start diversification scheme in order to achieve an efficient exploration process. To deal with the considered problem, a filter-set-based procedure is built in the FSA structure. Finally, an intensification scheme is applied as a final stage of the proposed method in order to overcome the slow convergence of SA-based methods.
SA [49]	The novel combination of the classical SA and fractional factorial analysis is termed the orthogonal SA herein. This study also introduces a dynamic penalty function to handle constrained optimization problems.
SA-GA [50]	In this paper, a novel adaptive real-parameter simulated annealing genetic algorithm (ARSAGA) that maintains the merits of genetic algorithm and simulated annealing is proposed. Adaptive mechanisms are also included to insure the solution quality and to improve the convergence speed.
PSO [51]	Unified Particle Swarm Optimization (UPSO) is a PSO scheme that harnesses the local and global variant of PSO, combining their exploration and exploitation abilities without imposing additional requirements in terms of function evaluations. For constrained problems a penalty function approach is employed and the algorithm is further modified to preserve feasibility of the encountered solutions.
PSO [52]	This paper presents an improved particle swarm optimizer (PSO) for solving mechanical design optimization problems involving problem-specific constraints and mixed variables such as integer, discrete and continuous variables. A constraint handling method called the 'fly-back mechanism' is introduced to maintain a feasible population. The standard PSO algorithm is also extended to handle mixed variables using a simple scheme.
EA^4 [53]	In this paper, a multiagent evolutionary algorithm called RAER based on the ERA multiagent modeling pattern is proposed, where ERA has the same architecture as Swarm including three parts of Environment, Reactive rules and Agents. RAER integrates a novel roulette inversion operator (RIO) proposed in this paper and theoretically proved to conquer the irrationality of the inversion operator (IO) designed by John Holland when used for real code stochastic optimization algorithms.
EA [54]	This paper introduces the notion of using co-evolution to adapt the penalty factors of a fitness function incorporated in a

	genetic algorithm (GA) for numerical optimization. The proposed approach produces solutions even better than those previously reported in the literature for other (GA-based and mathematical programming) techniques that have been particularly fine-tuned using a normally lengthy trial and error process to solve a certain problem or set of problems.
HS[5] [55]	In this paper a harmony search (HS) algorithm is conceptualized using the musical process of searching for a perfect state of harmony. It uses a stochastic random search instead of a gradient search so that derivative information is unnecessary.
IHS [56]	This study an Improved harmony search (IHS) algorithm employs a novel method for generating new solution vectors that enhances accuracy and convergence rate of harmony search (HS) algorithm. In this paper the impacts of constant parameters on harmony search algorithm are discussed and a strategy for tuning these parameters is presented.
HS-SQP[6] [57]	In Hybrid harmony search algorithm (HHSA) a sequential quadratic programming (SQP) is employed to speed up local search and improve precision of the HSA solutions. Moreover, an empirical study is performed in order to determine the impact of various parameters of the HSA on convergence behavior.
RA[7] [58]	In this study Random search method is applied to solve Welded beam problem.
SBM[8] [59]	This paper proposes a method for solving single objective constrained optimization problems by way of a socio-behavioural simulation model. The essence of the methodology is derived from the concept that the behaviour of an individual changes and improves due to social interaction with the society leaders. Leaders are identified after all individuals of a society are Pareto ranked according to constraint satisfaction. At the higher end, leaders of all societies interact among themselves for the overall improvement of the societies. Such overall improvement of individual societies leads to a better civilization.
SCA[9] [60]	This study is motivated from social interactions. The ability to mutually interact is a fundamental social behavior in all human and insect societies. Social interactions enable individuals to adapt and improve faster than biological evolution based on genetic inheritance alone. This is the driving concept behind the optimization algorithm introduced in this paper that makes use of the intra and intersociety interactions within a formal society and the civilization model to solve single objective constrained optimization problems. A society corresponds to a cluster of points in the parametric space while a civilization is a set of all such societies. Every society has its set of better performing individuals (leaders) that help others to improve through information exchange. This results in the migration of a point toward a better performing point, analogous to an intensified

	local search. Leaders improve only through an intersociety information exchange that results in the migration of a leader from a society to another. This helps the better performing societies to expand and flourish.
BFO[10] [61]	In this paper the basic BFOA is modified to search the solution regions. The author proposed four modifications to achieve this i.e.: (1) A single loop to include the chemotactic, reproduction and elimination-dispersal steps, (2) a definition of the step size values based on the features of the problem, (3) a constraint-handling mechanism and (4) a simple communication mechanism among bacteria to allow them to move towards promising regions of the search space.
DE[11] [62]	In this paper, the dynamic stochastic selection (DSS) is put forward within the framework of multimember differential evolution.
FA[12] [63]	In this study Firefly Algorithm (FA), is used for solving mixed continuous/discrete structural optimization problems. FA mimics the social behavior of fireflies based on their flashing characteristics.
ABC[13] [64]	In this study Simple Constrained Artificial Bee Colony, or SC-ABC as in the ABC for constrained problems, algorithm uses Deb's rules instead of the greedy selection in order to decide what solution will be kept. In the initialization phase only the first initialization of food sources is completely random. In other initialization phases the first new food source is the food source from the previous run of the algorithm which has the best fitness value. In other words, the runs of the SC-ABC algorithm are not completely independent. Therefore, exploitation of the good sources was increased. In order to increase the exploration the scout bee's phase was changed. In the scout phase the algorithm checks every possible solution. If the solution is not feasible, that food source is replaced with a new randomly produced solution.
PSOStr[65]	Particle Swarm Optimization Algorithm with Struggle Selection (PSOStr), is a hybrid of the evolutionary StrGA and the socially inspired PSOA. Struggle Genetic Algorithm (StrGA)+ Particle Swarm Optimization Algorithm (PSOA).
HSIA [66]	This paper presents a hybrid swarm intelligence approach (HSIA) for solving these nonlinear optimization problems which contain integer, discrete, zero-one and continuous variables. HSIA provides an improvement in global search reliability in a mixed-variable space and converges steadily to a good solution. An approach to handle various kinds of variables and constraints is discussed.
GA discrete-integer optimization	This paper describes the application of genetic algorithms to nonlinear constrained mixed discrete-integer optimization problems with optimal sets of parameters furnished by a meta-

[67] genetic algorithm.

GeneAs [68] In this paper, a combined genetic search technique (GeneAS) is suggested to solve mixed-integer programming problems often encountered in engineering design activities. GeneAS uses a combination of binary-coded and real-coded GAs to handle different types of variables. In handling discrete variables, GeneAS restricts its search only to the permissible values of the variable.

flc-aHGA [69] In this study an adaptive hybrid GA with an adaptive scheme using fuzzy logic controller fuzzy logic controller (flc-aHGA) is proposed to solve engineering design problems.

[1]Artificial Immune System;[2]Simulated Annealing;[3]Direct Search;[4]Evolutionary Algorithms;[5]Harmony Search;[6]Sequential Quadratic Programming;[7]Random Search;[8]Socio-Behavioral Model;[9]Society and Civilization Algorithm;[10]Bacterial Foraging Optimization;[11]Differential Evolution;[12]Firefly Algorithm;[13]Artificial Bee Colony. [Ref.]: Reference

6.3. Results analysis and discussions

The results obtained by DABC for solving welded beam problem are given in Table 3 and comparison of the result with the results reported in the literature are given in Table 4. The optimized best solution for the welded beam is h=0.20573; l=3.470489; t=9.036624; b=0.20573 and Cost is 1.724852. The numbers of evaluations taken by [41] are very less in comparison to all reported in the Table but the cost is comparatively on the higher side. In our case the cost is 1.7236, which is very close to the best solution given in the literature and only 38000 number of evaluations are taken to reach the best value, which demonstrates the better convergence than others presented in the Table 3.

The optimized results for the pressure vessel design problem are presented in Table 5 and Table 6 compares the result with the results reported in the literature, which also demonstrate that the proposed concept can improve the performance ABC because the cost calculated using ABC is quite higher than the one calculated using Firefly algorithm. Further it can be observed from the table that *PSO-GA* and *HS* violates the g_3 (third) constraint which explains that these methods are not the feasible one.

Finally the results for the helical compression string design are given in Table 7 and comparative results with those available in the literature are reported in Table 8. In this case also *PSO* violates the g8 constraint and hence provides infeasible solution, similarly *GA-FL* though corresponds to lowest cost but it also violates g_1 constraint, there again DABC emerges as a winner.

Analysis: From the above result discussion it can be analysed, that the proposed DABC taken less number of function evaluations to achieve the optimum results in comparison to the most of the algorithm referred for the comparisons. From this it can be analysed that the DABC converges faster towards the optimal solution.

Table 3. Statistical results of the DABC for the welded beam problem.

Best	Worst	Mean	S.D.	Avg. Generation
1.7296	1.9926	1.7236	1.62592	16.4

Table 4. Welded beam problem: comparison of DABC results with the results available in literature.

Algorithm [Ref.] ↓	*h*	*l*	*t*	*b*	Cost	No. of Evaluations
GA [42]	0.2489	6.1097	8.2484	0.2485	2.4000	6273
GA [43]	0.2088	3.4205	8.9975	0.2100	1.7483	N.A.
GA [44]	0.2489	6.1730	8.1789	0.2533	2.4331	320,080
GA [45]	0.2443	6.2117	8.3015	0.2443	2.3816	320,000
AIS[1]-GA [46]	0.2444	6.2183	8.2912	0.2444	2.3812	320,000
SA[2] [47]	0.2471	6.1451	8.2721	0.2495	2.4148.	N.A
SA-DS[3] [48]	0.2444	6.2158	8.2939	0.2444	2.3811	56,243
SA [49]	0.2444	6.2175	8.2915	0.2444	2.3810	N.A.
SA-GA [50]	0.2231	1.5815	12.8468	0.2245	2.2500	26,466
PSO [51]	N.A.	N.A.	N.A.	N.A.	1.9220	100,000
PSO [52]	0.2444	6.2175	8.2915	0.2444	2.3810	30,000
EA[4] [53]	0.2443	6.2201	8.2940	0.2444	2.3816	28,897
EA [54]	N.A.	N.A.	N.A.	N.A.	1.8245	N.A.
HS[5] [55]	0.2442	6.2231	8.2915	0.2443	2.381	110,000
HS [56]	0.2057	3.4705	9.0366	0.2057	1.7248	200,000
HS-SQP[6] [57]	0.2057	3.4706	9.0368	0.2057	1.7248	90,000
RA[7] [58]	0.2444	6.2819	8.2915	0.2444	2.3815	N.A.
SBM[8] [59]	0.2407	6.4851	8.2399	0.2497	2.4426	19,259
SCA[9] [60]	0.2444	6.2380	8.2886	0.2446	2.3854	33,095
BFO[10] [61]	0.2536	7.1410	7.1044	0.2536	2.3398	N.A.
DE[11] [62]	0.2444	6.2175	8.2915	0.2444	2.3810	24,000
FA[12] [63]	0.2015	3.562	9.0414	0.2057	1.73121	50,000
ABC[13] [64]	0.2055	3.471	9.0427	0.2058	1.72662	240000
DABC (*Present Study*)	0.2043	3.641	9.0522	0.2102	1.7236	38000

Table 5. Statistical analysis of the DABC for the pressure vessel problem.

Best	Worst	Mean	S.D.	Avg. Generation
5848.91	6071.77	5850.7610	0.0256683	11.5

Table 6. Pressure vessel design: Comparison of DABC results with the results available in literature.

Algorithm [Ref.] →	PSO-GA [65]	HS [56]	SA-DS[48]	FA [63]	ABC	Present Study (DABC)
f_{min}	5850.38306	5849.76169	5868.76484	5850.38306	5851.268058	5848.72
T_s	0.75	0.75	0.7683	0.75	0.7532	0.7510
T_h	0.375	0.375	0.3797	0.375	0.37500	0.73794
R	38.86010	38.86010	39.80962	38.86010	37.098187	37.0984
L	221.36549	221.36553	207.22555	221.36547	176.640750	176.637
g_1	-0.0000	-0.0000	-0.0000	-0.0000	-0.988451	-0.9874
g_2	-0.0043	-0.0043	-0.0000	-0.0043	-0.035883	-0.04165
g_3	**0.0446**[1]	**0.2713**	-10.7065	-0.0134	-5.297613	-5.19732
g_4	-18.6345	-18.6345	-32.7744	-18.6345	-33.359250	-19.9725

[1]Bold sets are violated sets.

Table 7. Statistical results of the DABC for the Helical compression spring design problem.

Best	Worst	Mean	S.D.	Avg. Generation
2.67153	3.9817	2.6586	12.8102	13.8

Table 8. Helical compression spring design: Comparison of DABC results with the results available in literature.

Algorithm [Ref.] →	N.A. [41]	PSO [66]	GA [67]	GA [68]	GA-FL[1] [69]	FA [63]	DABC (*Present Study*)
d	0.283	0.283	0.283	0.283	0.263	0.283	0.283
D	1.180701	1.223	1.227411	1.226	1.1096	1.223049	1.2173
N	10	9	9	9	9	9	9
g_1	-5430.9	-1008.81	-550.993	-713.51	**25154.82**	-1008.02	-913.091
g_2	-8.8187	-8.946	-8.9264	-8.933	-9.1745	-8.946	-8.781
g_3	-0.08298	-0.083	-0.0830	-0.083	-0.063	-0.083	-0.079
g_4	-1.8193	-1.77696	-1.7726	-1.491	-1.890	-1.777	-1.832
g_5	-1.1723	-1.3217	-1.3371	-1.337	-1.219	-1.322	-1.352
g_6	-5.4643	-5.4643	-5.4485	-5.461	-5.464	-5.464	-5.446
g_7	0	0	0	0	0	0	0
g_8	0.0000	**0.0001**	-0.0134	-0.0090	-0.0014	0.0000	0.0000
f_{min}	2.7995	2.659	2.6681	2.665	2.0283	2.6586	2.6452
No. of Eval.	N.A.	N.A.	N.A.	N.A.	100,000	50,000	32000

[1]Hybrid GA with fuzzy logic; Bold sets are violated sets.

7. Conclusions

In the present study, we proposed a concept of forward and backward movement in the ABC algorithm for locating an optimal solution. The proposed strategy though apparently simple was efficient when applied to three real life engineering design optimization problems with constraints containing discrete and continuous variables. The algorithm showed a good performance when compared to the results available in the literature. In future we plan to work for problems with larger dimensions and multi-objective optimization problems.

Acknowledgment

Authors are highly thankful to the unknown reviewers for providing necessary and valuable suggestions to improve the presentation of the paper.

References

1. Karaboga, D. (2005). An idea based on bee swarm for numerical optimization. *Technical report-TR06*, Erciyes University, Engineering Faculty, Computer Engineering Department.
2. Karaboga, D.; and 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.
3. Karaboga, D.; and Basturk, B. (2007). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 8(1), 687-697.

4. Sharma, T.K.; Pant, M.; and Abraham, A. (2011). Dichotomous search in ABC and its application in parameter estimation of software reliability growth models. *Proceedings of the World Congress on Nature & Biologically Inspired Computing (IEEE NaBIC)*, Spain, 214-219.
5. Karaboga, D.; and Basturk B. (2007(c)). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. *Proceedings of the Advances in Soft Computing-Foundations of Fuzzy Logic and Soft Computing. Springer- Verlag*, 4529, 789-798.
6. Baykasoglu, A.; Ozbakir L.; and Tapkan P. (2007). *Swarm intelligence focus on ant and particle swarm optimization, Artificial bee colony algorithm and its application to generalized assignment problem*. I-Tech Education and Publishing, Vienna, Austria, 113-144.
7. Pan, Q.-K.; Tasgetiren, M.F.; Suganthan, P.N.; and Chua, T.J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, 181(12), 2455-2468.
8. Karaboga, D.; and Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, 214(1), 108-132.
9. Tsai, P.-W. ; Pan, J.-S.; Liao, B.-Y.; and Chu, S.-C. (2009). Enhanced artificial bee colony optimization. *International Journal of Innovative Computing, Information and Control*, 5(12(B)), 5081-5092.
10. Haijun, D.; and Qingxian, F. (2009). Artificial bee colony algorithm based on Boltzmann selection strategy. *Computer Engineering and Applications*, 45(32), 53-55.
11. Kang, F.; Li, J.; and Xu, Q. (2009). Structural inverse analysis by hybrid simplex artificial bee colony algorithms. *Computers and Structures,* 87(13-14), 861-870.
12. Alatas, B. (2010). Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications,* 37(8), 5682-5687.
13. Zhu, G.; and Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, 217(7), 3166-3173.
14. Lei, X.; Huang, X.; and Zhang, A. (2010). Improved artificial bee colony algorithm and its application in data clustering. *Proceedings of the IEEE fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA)*, Changsha, China, 514-521.
15. Banharnsakun, A.; Achalakul, T.; and Sirinaovakul, B. (2011). The best-so-far selection in artificial bee colony algorithm. *Applied Soft Computing,* 11(2), 2888-2901.
16. Tuba, M.; Bacanin, N.; and Stanarevic, N. (2011). Guided artificial bee colony algorithm. *Proceedings of the European computing conference* (ECC'11), Paris, France, 398-403.
17. Li, G.; Niu, P.; and Xiao, X. (2012). Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. *Applied Soft Computing*, 12(1), 320-332
18. Gao, W.; and Liu, S. (2011). Improved artificial bee colony algorithm for global optimization. *Information Processing Letters*, 111(17), 871-882.

19. Bolaji, A.L.; Khader, A.T.; Al-Betar M.A.; and Awadallah, M.A. (2011). An improved artificial bee colony for course timetabling. *Proceedings of the Sixth International Conference on Bio-Inspired Computing: Theories and Applications*, Penang, Malaysia, 9-14.
20. Öner, A.; and Dengi, S.Ö.D. (2011). Optimization of university course scheduling problem with a hybrid artificial bee colony algorithm. *Proceedings of the IEEE Congress on Evolutionary Computation,* (CEC 2011)*,* New Orleans, LA, USA, 339-346.
21. Zhang, R.; Song, S.; and Wu, C. (2013). A hybrid artificial bee colony algorithm for the job shop scheduling problem. *International Journal of Production Economics*, 141(1), 167-178.
22. Kashan, M.H.; Nahavandi, N.; and Kashan, A.H. (2012). Disabc: A new artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, 12(1), 342-352.
23. Karaboga, D.; and Ozturk, C. (2009). Neural networks training by artificial bee colony algorithm on pattern classification. *Neural Network World,* 19(3), 279-292.
24. Shah, H.; Rozaida, G.; and Mohd Nawi, N. (2013). Global artificial bee colony algorithm for Boolean function classification. *Proceedings of the 5th Asian Conference on Intelligent Information and Database Systems,* ACIIDS 2013, Kuala Lumpur, Malaysia, 12-20.
25. Yeh, W.-C.; and Hsieh, T.-J. (2012). Artificial bee colony algorithm-neural networks for s-system models of biochemical networks approximation. *Neural Computing and Applications*, 21(2), 365-375.
26. Rao, R.S.; Narasimham, S.V.L.; and Ramalingaraju, M. (2008). Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm. *International Journal of Electrical Power and Energy Systems Engineering*, 1(2), 116-122.
27. Singh, A. (2009). An artificial bee colony algorithm for the leaf constrained minimum spanning tree problem. *Applied Soft Computing,* 9(2), 625-631.
28. Hu, Z.-H.; and Zhao M. (2009). Simulation on trav eling salesman problem (TSP) based on artificial bees colony algorithm. *Transactions of Beijing Institute of Technology*, 29(11), 978-982.
29. Kurban, T.; and Besdok, E. (2009). A comparison of RBF neural network training algorithms for inertial sensor based terrain classification. *Sensors,* 9(8), 6312-6329.
30. Zhang, C.S.; Ouyang, D.T.; and Ning, J.X. (2010). An artificial bee colony approach for clustering. *Expert Systems with Application,* 37(7), 4761-4767.
31. Karaboga, D.; Gorkemli, B.; Ozturk, C.; and Karaboga, N. (2011). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, 42(1), 21-57.
32. Yildiz, A.R. (2013) Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach. *Information Sciences*, 220, 399-407.
33. Wang, X.; Xie, X.; and Cheng, T.C.E. (2013). A modified artificial bee colony algorithm for order acceptance in two-machine flow shops. *International Journal of Production Economics*, 141(1), 14-23.

34. Garg, H.; Rani, M; and Sharma, S.P. (2013). Predicting uncertain behavior of press unit in a paper industry using artificial bee colony and fuzzy Lambda–Tau methodology. *Applied Soft Computing,* 13(4), 1869-1881.
35. Deng, G.; Xu, Z.; and Gu, X. (2012). A discrete artificial bee colony algorithm for minimizing the total flow time in the blocking flow shop scheduling. *Chinese Journal of Chemical Engineering*, 20(6), 1067-1073.
36. Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4), 311-338.
37. Ahandani, M.A.; Shirjoposh, N.P.; and Banimahd, R. (2010). Three modified versions of differential evolution algorithm for continuous optimization. *Soft Computing*, 15(4), 803-830.
38. Yang, X.-S. (2008). *Nature-inspired metaheuristic algorithms*. Luniver Press.
39. Cagnina, L.C.; Esquive, S.C.; and Coello, C.A.C. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica*, 32(3), 319-326.
40. Yildiz, A.R. (2008). Hybrid Taguchi: Harmony Search Algorithm for Solving Engineering Optimization Problems. *International Journal of Industrial Engineering*, 15(3), 286-293.
41. Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, 112(2), 223-229.
42. Leite, J.P.B.; and Topping, B.H.V. (1998). Topping BHV.: Improved genetic operators for structural engineering optimization. *Advances in Engineering Software*, 29(7-9), 529-562.
43. Coello, C.A.C. (2000). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems,* 17(4), 319-346.
44. Deb, K. (1991). Optimal design of a welded beam via genetic algorithms. *AIAA Journal*, 29(11), 2013-2015.
45. Lemonge, A.C.C.; and Barbosa, H.J.C. (2004). An adaptive penalty scheme for genetic algorithms in structural optimization. *International Journal for Numerical Methods in Engineering*, 59(5), 703-736.
46. Bernardino, H.S.; Barbosa, I.J.C.; and Lemonge, A.C.C. (2007). A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. *Proceedings of the IEEE Congress on Evolutionary Computation*, Singapore, 646-653.
47. Atiqullah, M.M; and Rao, S.S. (2000). Simulated annealing and parallel processing: an implementation for constrained global design optimization. *Engineering Optimization*, 32(5), 659-685.
48. Hedar, A.R.; and Fukushima, M. (2006). Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, 35(4), 521-649.
49. Liu, J.-L. (2005). Novel orthogonal simulated annealing with fractional factorial analysis to solve global optimization problems. *Engineering Optimization*, 37(5), 499-519.

50. Hwang, S.-F.; and He, R.-S. (2006). A hybrid real-parameter genetic algorithm for function optimization. *Advanced Engineering Informatics*, 20(1), 7-21.
51. Parsopoulos, K.E.; and Vrahatis, M.N. (2005). Unified particle swarm optimization for solving constrained engineering optimization problems. *Proceedings of the ICNC 2005*, Changsha, China, 3612, 582-591.
52. He, S.; Prempain, E.; and Wu, Q.H. (2004). An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, 36(5), 585-605.
53. Zhang, J.; Liang, C.; Huang, Y.; Wu, J.; and Yang, S. (2009). An effective multiagent evolutionary algorithm integrating a novel roulette inversion operator for engineering optimization. *Applied Mathematics and Computation*, 211(2), 392-416.
54. Coello, C.A.C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2), 113-127.
55. Lee, K.S.; and Geem, Z.W. (2004). A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36-38), 3902-3933.
56. Mahdavi, M.; Fesanghary, M.; and Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2), 1567-1579.
57. Fesanghary, M.; Mahdavi, M.; Minary-Jolandan, M.; and Alizadeh, Y. (2008). Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40), 3080-3091.
58. Siddall, J.N. (1972). *Analytical decision-making in engineering design*. Englewood Cliffs: Prentice-Hall.
59. Akhtar, S.; Tai, K.; and Ray, T. (2002). A socio-behavioural simulation model for engineering design optimization. *Engineering Optimization*, 34(4), 341-354.
60. Ray, T.; and Liew, K.M. (2003). Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, 7(4), 386-396.
61. Montes, E.M.; and Ocaña, B.H. (2009). Modified bacterial foraging optimization for engineering design. *Proceedings of the Artificial neural networks in engineering conference (ANNIE'2009), ASME Press series, Intelligent engineering systems through artificial neural networks*, St. Louis, Missouri, USA, 19, 357–364.
62. Zhang, M.; Luo, W.; and Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, 178(15), 3043-3074.
63. Gandomi, A.H.; Yang, X.-S.; and Alavi, A.H. (2011). Mixed variable structural optimization using Firefly Algorithm. *Computers and Structures*, 89(23-24), 2325-2336.
64. Brajevic, I.; Tuba, M.; and Subotic, M. (2011). Performance of the improved artificial bee colony algorithm on standard engineering constrained problems.

International Journal of Mathematics and Computers in Simulation, 5(2), 135-143.

65. Dimopoulos, G.G. (2007). Mixed-variable engineering optimization based on evolutionary and social metaphors. *Computer Methods in Applied Mechanics and Engineering*, 196(4-6), 803-817.
66. Guo, C.-X.; Hu, J.-S.; Ye, B.; and Cao, Y.-J. (2004). Swarm intelligence for mixed-variable design optimization. *Journal of Zhejiang University Science*, 5(7), 851-860.
67. Wu, S.-J.; and Chow, P.-E. (1995). Genetic algorithms for nonlinear mixed discrete-integer optimization problems via metagenetic parameter optimizations. *Engineering Optimization*, 24(2), 137–59.
68. Deb, K.; and Goyal, M.; (1997). Optimizing engineering designs using a combined genetic search. *Proceedings of the 7th international conference on genetic algorithms*, East Lansing, MI, USA, 512–28.
69. Yun, Y.S. (2005). *Study on adaptive hybrid genetic algorithm and its applications to engineering design problems*. M.Sc. Thesis. Waseda University, Shinjuku, Tokyo, Japan.