Applied Soft Computing 11 (2011) 3021–3031

Contents lists available at ScienceDirect

Applied Soft Computing

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

A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems

Dervis Karaboga*, Bahriye Akay

Erciyes University, The Department of Computer Engineering, 38039 Melikgazi, Kayseri, Turkiye

ARTICLE INFO

Article history:
Received 1 September 2008
Received in revised form 2 September 2010
Accepted 1 December 2010
Available online 17 December 2010

Keywords:
Swarm intelligence
Modified Artificial Bee Colony algorithm
Constrained optimization

ABSTRACT

Artificial Bee Colony (ABC) algorithm was firstly proposed for unconstrained optimization problems on where that ABC algorithm showed superior performance. This paper describes a modified ABC algorithm for constrained optimization problems and compares the performance of the modified ABC algorithm against those of state-of-the-art algorithms for a set of constrained test problems. For constraint handling, ABC algorithm uses Deb's rules consisting of three simple heuristic rules and a probabilistic selection scheme for feasible solutions based on their fitness values and infeasible solutions based on their violation values. ABC algorithm is tested on thirteen well-known test problems and the results obtained are compared to those of the state-of-the-art algorithms and discussed. Moreover, a statistical parameter analysis of the modified ABC algorithm is conducted and appropriate values for each control parameter are obtained using analysis of the variance (ANOVA) and analysis of mean (ANOM) statistics.

1. Introduction

Structural optimization, engineering design, VLSI design, economics, allocation and location problems are just a few examples of fields in which constrained optimization problems are encountered [1]. A constrained optimization problem (1) is defined as finding parameter vector $\vec{x}$ that minimizes an objective function $f(\vec{x})$ subject to inequality and/or equality constraints:

$$\begin{aligned} &\text{minimize } f(\vec{x}), \quad \vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n \\ &\qquad\qquad l_i \le x_i \le u_i, \qquad\qquad i = 1, \ldots, n \\ &\text{subject to :} \quad g_j(\vec{x}) \le 0, \qquad\qquad \text{for } j = 1, \ldots, q \\ &\qquad\qquad h_j(\vec{x}) = 0, \qquad\qquad \text{for } j = q+1, \ldots, m \end{aligned} \tag{1}$$

The objective function f is defined on a search space, $\mathbb{S}$, which is defined as a n-dimensional rectangle in $\mathbb{R}^n$ ($\mathbb{S} \subseteq \mathbb{R}^n$). Domains of variables are defined by their lower and upper bounds. A feasible region $\mathbb{F} \subseteq \mathbb{S}$ is defined by a set of m additional constraints ($m \ge 0$) and $\vec{x}$ is defined on feasible space ($\vec{x} \in \mathbb{F} \in \mathbb{S}$). At any point $\vec{x} \in \mathbb{F}$, constraints g_j that satisfy $g_j(\vec{x}) = 0$ are called active constraints at $\vec{x}$. By extension, equality constraints h_j are also called active at all points of $\mathbb{S}$ [2]. Constrained optimization problems are hard to optimization algorithms and also no single parameter (number of linear, nonlinear and active constraints, the ratio $\rho = |\mathbb{F}| / |\mathbb{S}|$, type of the function, number of variables) is proved to be significant as a major measure of difficulty of the problem [3].

Since most of the optimization algorithms have been primarily designed to address unconstrained optimization problems, constraint handling techniques are usually incorporated in the algorithms in order to direct the search towards the feasible regions of the search space. Methods dealing with the constraints were grouped into four categories by Koziel and Michalewicz [4]: (i) methods based on preserving feasibility of solutions by transforming infeasible solutions to feasible ones with some operators [5,6]; (ii) methods based on penalty functions which introduce a penalty term into the original objective function to penalize constraint violations in order to solve a constrained problem as an unconstrained one. [7–10]; (iii) methods that make a clear distinction between feasible and infeasible solutions [11–15]; (iv) other hybrid methods combining evolutionary computation techniques with deterministic procedures for numerical optimization [16–19].

Koziel and Michalewicz investigated a decoder based constraint handling approach which incorporates a homomorphous mapping, between n-dimensional cube and a feasible search space for solving constrained numerical optimization problems by evolutionary algorithms [4]. The method uses a decoder which transforms a constrained problem to an unconstrained problem via a homomorphous mapping (HM).

Another study related with evolutionary algorithms is Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) proposed by Hamida and Schoenauer for constrained optimization problems based on a population level adaptive

* Corresponding author. Tel.: +90 352 437 49 01x32577; fax: +90 352 437 57 84.
E-mail addresses: karaboga@erciyes.edu.tr (D. Karaboga), bahriye@erciyes.edu.tr (B. Akay).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.12.001

penalty function to handle constraints [20]. ASCHEA employs a constraint-driven mate selection for recombination and a segregational selection that favors a given number of feasible individuals and utilizes an equality constraint handling strategy which starts a large feasible domain and tightens it progressively [20].

Runarsson and Yao also investigated the performance of an evolution strategy using different constraint handling methods [21]. They described the over-penalty approach (OPA) which ranks feasible individuals according to their objective function value, followed by the infeasible solutions ranked according to penalty function value.

Runarsson and Yao introduced an approach which balances the objective and penalty functions by stochastic ranking (SR) in Evolution Strategy (ES) [22]. The approach avoids setting a hard-to-set parameter penalty factor and treats constrained optimization as multi-objective optimization where constraints are regarded as an additional objective function. Moreover, they improved the performance of the evolution strategy (Improved Stochastic Ranking, ISR) by employing a search mechanism to overcome the problem of a search bias aligned with the coordinate axis [21].

Mezura-Montes and Coello Coello proposed an evolution strategy which is called Simple multi-membered Evolution Strategy (SMES) to solve global nonlinear optimization problems [23]. The approach uses a diversity mechanism based on allowing infeasible solutions to remain in the population instead of using a penalty function. A feasibility-based comparison mechanism is used to guide the process toward the feasible region of the search space. Also, the initial step size of the evolution strategy is reduced in order to perform a finer search and a combined panmictic recombination technique is applied to improve its exploitation capabilities [23].

In this work, ABC algorithm described by Karaboga [24] inspiring from the foraging behaviour of honey bees is modified to solve constrained optimization problems and its performance is investigated on constrained problems. ABC algorithm is extended by replacing the selection mechanism of the simple ABC algorithm with Deb's selection mechanism [15] in order to cope with the constraints and introducing a probabilistic selection scheme that assigns probability values to feasible solutions based on their fitness values and to infeasible individuals based on their violations. Although it is a known fact that the handling technique employed may influence the performance of the algorithm, a handling technique using Deb's rules, in the category of methods searching for feasible solutions on assumption that any feasible solution is better than any infeasible one [15] was adopted, since it has advantages in terms of simplicity, computational cost, fine tuning requirement etc. over other techniques.

The performance of ABC algorithm has been tested on 13 well-known constrained optimization test problems described in [22] and compared with those of other previous studies [4,20–23].

The remainder of the paper is organized as follows. In Section 2, the modified ABC algorithm adapted for solving constrained optimization problems is introduced and then in Section 3, the proposed algorithm is compared with other algorithms. Finally, in Section 4, conclusions and future works are provided.

2. Modified Artificial Bee Colony algorithm

Artificial Bee Colony algorithm is a recently proposed optimization algorithm that simulates the foraging behaviour of a bee colony [24]. In a real bee colony there are some tasks done by specialized individuals. Bees try to maximize the nectar amount unloaded to the food stores in the hive by this division of labour and self-organization. Division of labor and self-organization are essential components of swarm intelligence. The minimal model of swarm-intelligent forage selection in a honey bee swarm consists of three kinds of bees: employed bees, onlooker bees, and scout bees [25]. Employed bees are responsible from exploiting the nectar sources explored before and they give information to the other waiting bees in the hive about the quality of the food source which they are exploiting. Onlooker bees wait in the hive and establish food source to exploit depending on the information shared by the employed bees. Scouts search environment in order to find a new food source depending on an internal motivation or external clues or randomly.

ABC algorithm uses this minimal model to simulate the collective intelligence in foraging behaviour of a honey bee swarm. In ABC algorithm, each food source is exploited by only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. Half of the colony comprises employed bees and the other half includes the onlooker bees. The employed bee whose food source has been abandoned by its bee becomes a scout. In ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees or the onlooker bees is equal to the number of food sources in the population.

Pseudo-code of the modified ABC algorithm proposed for solving constrained problems is given in Algorithm 1:

Algorithm 1 *(Pseudo-code of main body of ABC algorithm).*

```
Initialization
Evaluation
cycle = 1
repeat
    Employed Bees Phase
    Calculate Probabilities for Onlookers
    Onlooker Bees Phase
    Scout Bees Phase
    Memorize the best solution achieved so far
    cycle = cycle + 1
until cycle = Maximum Cycle Number
```

In the first step, ABC algorithm generates a randomly distributed initial population of $sn/2$ solutions (food source positions), where sn denotes the size of population. Each solution $\vec{x}_i$ ($i = 1, 2, \ldots, sn/2$) is a D-dimensional vector. Here, D is the number of optimization parameters. Because initialization with feasible solutions is very time consuming process and in some cases it is impossible to produce a feasible solution randomly, ABC algorithm does not consider the initial population to be feasible. In initialization phase, random values between the lower and the upper boundaries of the parameters are assigned for the parameters of solutions. $failure_i$ is the non-improvement number of the solution $\vec{x}_i$, used for the abandonment. Initialization procedure is given in Algorithm 2.

Algorithm 2 *(Initialization step of ABC algorithm).*

```
for i = 1 to sn/2 do
    for j = 1 to D do
        Generate x_i solution
        x_i^j = x_min^j + rand(0,1)(x_max^j − x_min^j)        (2)
        where x_min^j and x_max^j are lower and upper bound of the
    parameter j, respectively.
    end for
    failure_i = 0
end for
```

After initialization, the population is evaluated and is subjected to repeated cycles of the search processes of the employed bees, the onlooker bees and scout bees. Employed bee procedure of the ABC algorithm is presented in Algorithm 3.

Algorithm 3 *(Employed bees phase of ABC algorithm).*

1: **for** $i = 1$ to $sn/2$**do**
2: **for** $j = 1$ to D**do**
3: Produce a new food source $\vec{v}_i$ for the employed bee of the food source $\vec{x}_i$ by using (3)

$$v_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), & \text{if } R_j < MR \\ x_{ij}, & \text{otherwise} \end{cases} \quad (3)$$

where $k \in \{1, 2, \ldots, sn\}$ is randomly chosen index that has to be different from i and ϕ_{ij} is uniformly distributed random real number in the range of $[-1,1]$. R_j is uniformly distributed random real number in the range of [0,1] and MR is a control parameter of ABC algorithm in the range of [0,1] which controls the number of parameters to be modi?ed.
4: **end for**
5: If no parameter is changed, change one random parameter of the solution $\vec{x}_i$ by (4)

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (4)$$

where j is uniformly distributed random integer number in the range $[1, D]$.
6: Evaluate the quality of $\vec{v}_i$
7: Apply the selection process between $\vec{v}_i$ and $\vec{x}_i$ based on Deb's method
8: If solution $\vec{x}_i$ does not improve $failure_i$ = $failure_i$ + 1, otherwise $failure_i$ = 0
9: **end for**

An employed bee produces a modification (3) on the position of the food source (solution) in her memory depending on the local information (visual information) and evaluates the nectar amount (fitness value, quality) of the new source (new solution). As can be seen from Eq. (3), as the difference between the parameters of the x_{ij} and x_{kj} decreases, the perturbation on the position x_{ij} decreases. Thus, as the search approaches to the optimum solution in the search space, the step length is adaptively reduced. The first modification of ABC algorithm arises here. In the unconstrained version of ABC algorithm, only one randomly chosen parameter x_{ij} is modified while other parameters except for x_{ij} are copied from the old solution $\vec{x}_i$. However, in the modified ABC algorithm, for each parameter x_{ij}, a uniformly distributed random real number, $(0 \le R_j \le 1)$, is produced and if the real number is less than the control parameter value, modification rate (MR), and the parameter x_{ij} is modified. Here, we ensure at least one parameter to be modified if no parameter is changed.

After producing a new food source (solution), ABC algorithm makes a selection. The second modification of ABC algorithm to solve constrained optimization problems is using Deb's rules instead of using greedy selection in selection process. Structure of the algorithm directs the solutions to feasible region in running process due to the Deb's rules employed instead of greedy selection. Applying Deb's rules, the bee either memorizes the new position by forgetting the old one or keeps the old one. Deb's method uses a tournament selection operator, where two solutions are compared at a time by applying following criteria:

- Any feasible solution ($violation_i \le 0$) is preferred to any infeasible solution ($violation_j > 0$) (solution i is dominant);
- Among two feasible solutions($violation_i \le 0$, $violation_j \le 0$), the one having better objective function value is preferred ($f_i < f_j$, solution i is dominant);
- Among two infeasible solutions ($violation_i > 0$, $violation_j > 0$), the one having smaller constraint violation is preferred ($violation_i < violation_j$, solution i is dominant).

After all employed bees complete the search process, they share the nectar information of the food sources and their position information with the onlooker bees on the dance area by calculating probability values. This process is simulated by the procedure given in Algorithm 4.

Algorithm 4 *(Calculate probabilities for onlookers).*

1: **for** $i = 1$ to $sn/2$**do**
2: Calculate the probability values p_i for the solutions using fitness and/or violation of the solutions by (5)

$$p_i = \begin{cases} 0.5 + \left(\dfrac{fitness_i}{\sum_{j=1}^{sn} fitness_j} \right) \times 0.5 & \text{if solution is feasible} \\ \left(1 - \dfrac{violation_i}{\sum_{j=1}^{sn} violation_j} \right) \times 0.5 & \text{if solution is infeasible} \end{cases} \quad (5)$$

where $violation_i$ is the penalty value of the solution $\vec{x}_i$ and $fitness_i$ is the fitness value of the solution $\vec{x}_i$ which is proportional to the nectar amount of that food source. The fitness is determined by (6).

$$fitness_i = \begin{cases} 1/(1 + f_i) & \text{if } f_i \ge 0 \\ 1 + abs(f_i) & \text{if } f_i < 0 \end{cases} \quad (6)$$

where f_i is the cost value of the solution $\vec{x}_i$.
3: **end for**

Since infeasible solutions are allowed to populate in the colony, a third modification is needed here to assign probability values for infeasible solutions as well as for feasible ones. Probability values of infeasible solutions are between 0 and 0.5 while those of feasible ones are between 0.5 and 1. By a selection mechanism like roulette wheel, solutions are selected probabilistically proportional to their fitness values incase of feasible solutions and inversely proportional to their violation values incase of infeasible solutions.

An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount. As in the case of the employed bee, onlooker bee produces a modification on the position of the food source (3) selected based on the probability value and checks its nectar amount. If a parameter value produced by this operation exceeds its predetermined boundaries, the parameter can be set to an acceptable value. In this work, the value of the parameter exceeding its boundary is set to its boundaries. Onlooker stage is conducted by pseudo-code block given in Algorithm 5.

Algorithm 5 *(Onlooker bees phase of ABC algorithm).*

1: $t = 0, i = 1$
2: **repeat**
3: **if** $random < p_i$**then**
4: $t = t + 1$
5: **for** $j = 1$ to D**do**
6: Produce a new food source $\vec{v}_i$ for the onlooker bee of the food source $\vec{x}_i$ by using (3)
7: **end for**
8: Apply the selection process between $\vec{v}_i$ and $\vec{x}_i$ based on Deb's method
9: If solution $\vec{x}_i$ does not improve $failure_i = failure_i + 1$, otherwise $failure_i = 0$
10: **end if**
11: $i = i + 1$
12: $i = i \bmod ((sn/2) + 1)$
13: **until** $t = sn/2$

After all onlookers are distributed, food sources that are not worth to exploit anymore are determined. If a solution can not be improved through a predetermined number of cycles ("*limit*"), it is decided to be abandoned. The food source abandoned by the bees is replaced with a new food source discovered by the scouts. This is simulated by producing a position randomly (2) and replacing it with the abandoned one. Another difference between the versions of ABC proposed for unconstrained and constrained problems is that artificial scouts are produced at a predetermined period of

cycles for discovering new food sources randomly. This period is another control parameter of the modified algorithm called scout production period (*SPP*). At each *SPP* cycle, it is controlled if there is an abandoned food source exceeding *limit* or not. If there is, a scout production process is carried out. Scout production process of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be in the population. Scout production process is given inAlgorithm 6.

Algorithm 6 *(Scout bees phase of ABC algorithm).*

```
if cycle mod SPP = 0 then
  if max(failure_i) > limit then
    Replace x̄_i with a new randomly produced solution by (2)
  end if
end if
```

In overall, the modified ABC algorithm employs two more control parameters in order to improve the convergence capability of ABC algorithm for constrained optimization problems. These are *MR* and *SPP* parameters. Another modification is using a selection approach based on Deb's three domination rules instead of greedy selection. Furthermore, for allowing infeasible solutions in the population to provide diversity, infeasible solutions are assigned probability values inversely proportional to their constraint violations, which gives them chance to be selected in roulette wheel selection-like onlooker allocation process.

3. Experimental study and discussion

In order to evaluate the performance of the modified ABC algorithm, we used a set of 13 benchmark problems described in [22]. This set includes various forms of objective function such as linear, nonlinear and quadratic. Type of objective function, the number of linear equalities (LE), nonlinear equalities (NE), linear inequalities (LI), nonlinear inequalities (NI), and variable size of each case (D) are given in Table 1. In Table 1, ρ is an estimate of the ratio between the feasible region and the entire search space computed by $\rho = |\mathbb{F}|/|\mathbb{S}|$ where $|\mathbb{F}|$ is the number of feasible solutions and $|\mathbb{S}|$ is the total number of solutions randomly generated [2]. The results of ABC algorithm is compared with the results of other methods in relevant literature in order to make fair comparison including homomorphous mapping (HM) method [4], stochastic ranking (SR) method [22], improved stochastic ranking (ISR) method [21], over-penalty approach (OPA) [21], adaptive segregational constraint handling evolutionary algorithm (ASCHEA) [20], simple multi-membered evolution strategy (SMES) [23], genetic algorithm (GA) [23], differential evolution (DE), and particle swarm optimization (PSO) algorithm [26].

3.1. Settings

HM employs a decoder-based constrained handling method. It uses Gray coding for representation. Each variable is represented with 25 bits. The algorithm incorporates proportional selection (no elitism), function scaling, and standard operators (flip mutation and 1-point crossover). Population size is 70, crossover rate is 0.9, generation gap is 100%, and maximum number of generations is 20,000, so HM performs 1,400,000 objective function evaluations [4].

SR uses a constraint-handling technique based on the stochastic ranking scheme in evolution strategy. A (30,200)-ES is used and all runs are terminated after 1750 generations, so it performs $200 \times 1750 = 350,000$ objective function evaluations. All equality constraints have been converted into inequality constraints, $|h_j| \leq \epsilon$ with $\epsilon = 0.0001$ [22].

ISR uses (60,400) evolution strategy through 875 generation, i.e. $400 \times 875 = 350,000$ objective function evolution. The improvement of the algorithm is in the constraint handling method. This method uses a γ parameter which is used for scaling step length. A step length reduction of around 0.85 is used.

OPA uses the (60,400) evolution strategy with a constraint handling method based on penalty approach. All runs are terminated after 875 generations except for g12, which is run for 87 generations, in this way the number of function evaluations is equivalent $400 \times 875 = 350,000$ [21].

ASCHEA employs a penalty based constrained handling method. In ASCHEA, the tolerance value ϵ is varied with respect to the generation number. ASCHEA uses (100 + 300)-ES segregational selection strategy. Mutation is the standard Gaussian mutation and self-adaptive standard deviations, with initial values of 0.03, global learning rate of 0.1 and local learning rate of 1. Recombination was the standard arithmetical crossover. The crossover and mutation rates are set to 0.9 and ASCHEA performs 1,500,000 evaluations [20].

GA uses Deb's rules as constraint handling method. All equality constraints have been converted into inequality constraints, $|h_j| \leq \epsilon$ with ϵ varying dynamically. For GA, population size is 200, maximum number of generations is 1200, crossover rate is 0.8, mutation rate is 0.6 and the number of objective function evaluations is 240,000 [23].

SMES uses Debs rules for constraint handling and (100,300)-ES. The number of generations is 800 and the total objective function evaluations is 240,000 [23].

The PSO algorithm employs Deb's rules for constraint handling. The swarm size is 50 and the generation number is 7000. Hence, PSO performs 350,000 objective function evaluations. Cognitive and social components are both set to 1. Inertia weight is uniform random real number in the range [0.5,1]. All equality constraints are converted into inequality constraints, $|h_j| \leq \epsilon$ with $\epsilon = 0.001$ [26].

In the DE algorithm, F is a real constant which affects the differential variation between two solutions and set to 0.5 in our experiments. Value of crossover rate that controls the change of the diversity of the population is chosen to be 0.9 as recommended in [27]. Population size is 40, maximum generation number is 6000 and it uses Deb's rules.

Table 1
Values of ρ for the 13 test problems. D: dimension of the problem; LI: linear inequality; NI: nonlinear inequality; LE: linear equality; NE: nonlinear equality [23].

	D	Type of Prob.	ρ	LI	NI	LE	NE
g01	13	Quadratic	0.0003%	9	0	0	0
g02	20	Nonlinear	99.9973%	1	1	0	0
g03	10	Nonlinear	0.0026%	0	0	0	1
g04	5	Quadratic	27.0079%	0	6	0	0
g05	4	Nonlinear	0.0000%	2	0	0	3
g06	2	Nonlinear	0.0057%	0	2	0	0
g07	10	Quadratic	0.0000%	3	5	0	0
g08	2	Nonlinear	0.8581%	0	2	0	0
g09	7	Nonlinear	0.5199%	0	4	0	0
g10	8	Linear	0.0020%	3	3	0	0
g11	2	Quadratic	0.0973%	0	0	0	1
g12	3	Quadratic	4.7697%	0	9^3	0	0
g13	5	Nonlinear	0.0000%	0	0	1	2

Table 2
Statistical results obtained by ABC algorithm for 13 test functions over 30 independent runs using 240,000 objective function evaluations.

Problem	Optimal	Best	Mean	Worst	S.D.
g01	−15.000	−15.000	−15.000	−15.000	0.000
g02	0.803619	0.803598	0.792412	0.749797	0.012
g03	1.000	1.000	1.000	1.000	0.000
g04	−30665.539	−30665.539	−30665.539	−30665.539	0.000
g05	5126.498	5126.484	5185.714	5438.387	75.358
g06	−6961.814	−6961.814	−6961.813	−6961.805	0.002
g07	24.306	24.330	24.473	25.190	0.186
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.634	680.640	680.653	0.004
g10	7049.25	7053.904	7224.407	7604.132	133.870
g11	0.75	0.750	0.750	0.750	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.760	0.968	1.000	0.055

In ABC algorithm, the value of modification rate (*MR*) is 0.8, colony size (*sn*) is 40 and the maximum cycle number (*MCN*) is 6000. So, the total objective function evaluation number is 240,000 as in the SMES, GA and DE. The value of limit is equal to $0.5 \times sn \times D$ where D is the dimension of the problem and sn is the number of solutions in the population, and *SPP* is also $0.5 \times sn \times D$. Experiments were repeated 30 times each starting from a random population with different seeds.

3.2. Results and discussion

Experimental results of ABC algorithm are given in Table 2. As seen from Table 2, although ABC algorithm has found the global minimum of the seven problems (g01, g03, g04, g06, g08, g11, g12) through 240,000 cycles. ABC was able to find close results to global optimum on five of rest functions (g02, g05, g07, g09, g10). However, on one problem, g13, ABC algorithm could not find the global optimum in the specified maximum number of cycles. Comparative results of the best and mean solutions of the investigated algorithms are presented in Tables 3–5. From Table 3, according to the best results, it can be seen that ABC performs better than HM, ASCEHA, GA, PSO; ABC and SR show equal performances; ISR, OPA, SMES and DE demonstrate better performance than ABC does. However, in terms of the mean results presented in Tables 4 and 5, ABC is also better than SR, SMES and DE algorithms beside HM, ASCHEA, GA and PSO. While OPA is better than ABC with respect to the best results, ABC and OPA show similar performances according to the mean results. It means that ABC is more robust than SR, DE, SMES and OPA algorithms. ISR is better than ABC in both cases. Here, it should be noted that the results of OPA, SR, ISR and PSO algorithms were obtained after 350,000 function evaluations while those of DE, GA, SMES and ABC were obtained after 240,000 function evaluations. As mentioned earlier, on constrained optimization problems, no single parameter (number of linear, nonlinear, active constraints, the ratio $\rho = |\mathbb{F}| / |\mathbb{S}|$, type of the function, number of variables) is proved to be significant as a major measure of difficulty of the problem [3].

Since the results belonging to OPA, SR, ISR and PSO were produced for 350,000 objective function evaluations (colony size=40, cycle=8750), the statistics for 30 runs of ABC algorithm using the same number of total evaluation were produced and presented in Table 6. From Tables 4 and 6, the success rates of algorithms obtained by comparing the results of algorithms and the ABC algorithm dually are given in Table 7. As seen from Table 6, when ABC algorithm is run more cycles, the performance of the algorithm improves in terms of best, mean, worst solutions and standard deviation. It should be mentioned that, in SMES, a high tolerance value, ϵ, is applied and then it is gradually decreased to a lower value. Also, the range of ϵ is tuned with respect to the problem. Hence it requires a priori knowledge about the optimization problem. In ABC algorithm, such a mechanism is not used. ABC algorithm employs the same value of ϵ for all problems during the optimization process. In other words, ABC algorithm is much simpler than SMES and does not require priori knowledge for determining the range of ϵ.

In order to analyze on which specific kind of problem, the modified ABC algorithm performs better or not, the results should be examined. Modified ABC algorithm cannot find the optimum solution for g05, g10 and g13 for each run. g05 and g13 are nonlinear

Table 3
The best solutions obtained by the HM, SR, ISR, OPA, ASCHEA, SMES, GA, DE, PSO and ABC algorithms for 13 test functions over 30 independent runs. (–) Means that no feasible solutions were found. Na: not available.

P	Optimal	HM [a] [4]	SR [b] [22]	ISR [b] [21]	OPA [c] [21]	ASCHEA [d] [20]	GA [e] [23]	SMES [e] [23]	PSO [e] [26]	DE [e]	ABC [e]
g01	−15.000	−14.7864	−15.000	−15.000	−15.000	−15.0	14.440	−15.000	15.000	−15.000	−15.000
g02	0.803619	0.79953	0.803515	0.803619	0.803619	0.785	0.796231	0.803601	0.669158	0.472	0.803598
g03	1.000	0.9997	1.000	1.001	0.747	1.0	0.990	1.000	0.993930	1.000	1.000
g04	−30665.539	−30664.5	−30665.539	−30665.539	−30665.539	−30665.5	−30626.053	−30665.539	−30665.539	−30665.539	−30665.539
g05	5126.498	–	5126.497	5126.497	5126.497	5126.5	–	5126.599	5126.484	5126.484	5126.484
g06	−6961.814	−6952.1	−6961.814	−6961.814	−6961.814	−6961.81	−6952.472	−6961.814	−6161.814	−6954.434	−6961.814
g07	24.306	24.620	24.307	24.306	24.306	24.3323	31.097	24.327	24.370153	24.306	24.330
g08	0.095825	0.0958250	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.63	680.91	680.630	680.630	680.630	680.630	685.994	680.632	680.630	680.630	680.634
g10	7049.25	7147.9	7054.316	7049.248	7049.248	7061.13	9079.770	7051.903	7049.381	7049.248	7053.904
g11	0.75	0.75	0.750	0.750	0.750	0.75	0.75	0.75	0.749	0.752	0.750
g12	1.000	NA	1.000	1.000	1.000	NA	1.000	1.000	1.000	1.00	1.000
g13	0.053950	NA	0.053957	0.053942	0.447118	NA	0.134057	0.053986	0.085655	0.385	0.760

[a] Using decoder-based.
[b] Using stochastic ranking.
[c] Using over-penalty approach.
[d] Using penalty-based.
[e] Using Deb's method.

Table 4
The mean solutions obtained by the HM, SR, ISR, OPA, ASCHEA, SMES, GA, DE, PSO and ABC algorithms for 13 test functions over 30 independent runs and total success numbers of algorithms. A result in boldface indicates a better result or that the global optimum (or best known solution) was reached. Success number (number of results in boldface) indicates that on how many problems the algorithm reached the optimum or did the best. (–) Means that no feasible solutions were found. Na: Not available.

P	Optimal	HM [a] [4]	SR [b] [22]	ISR [b] [21]	OPA [c] [21]	ASCHEA [d] [20]	GA [e] [23]	SMES [e] [23]	PSO [e] [26]	DE [e]	ABC [e]
g01	−15.000	−14.7082	**−15.000**	**−15.000**	**−15.000**	−14.84	−14.236	**−15.000**	−14.710	−14.555	**−15.000**
g02	0.803619	**0.79671**	0.781975	0.782725	0.776283	0.59	0.788588	0.785238	0.419960	0.665	0.792412
g03	1.000	0.9989	1.000	**1.001**	0.257	0.99989	0.976	1.000	0.764813	1.000	1.000
g04	−30665.539	−30655.3	**−30665.539**	**−30665.539**	**−30665.539**	−30665.5	−30590.455	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**
g05	5126.498	–	5128.881	**5126.497**	5268.610	5141.65	–	5174.492	5135.973	5264.270	5185.714
g06	−6961.814	−6342.6	−6875.940	**−6961.814**	**−6961.814**	−6961.81	−6872.204	−6961.284	**−6961.814**	–	−6961.813
g07	24.306	24.826	24.374	**24.306**	24.307	24.6636	34.980	24.475	32.407	24.310	24.473
g08	0.095825	0.0891568	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0.095799	**0.095825**	**0.095825**	**0.095825**	**0.095825**
g09	680.63	681.16	680.656	**680.630**	**680.630**	680.641	692.064	680.643	680.630	**680.630**	680.640
g10	7049.25	8163.6	7559.192	7049.250	**7049.248**	7497.434	10003.225	7253.047	7205.5	7147.334	7224.407
g11	0.75	**0.75**	**0.750**	**0.750**	0.756	**0.75**	**0.75**	**0.75**	0.749	0.901	**0.750**
g12	1.000	NA	**1.000**	**1.000**	0.999889	NA	**1.000**	**1.000**	0.998875	**1.000**	**1.000**
g13	0.053950	NA	**0.057006**	0.066770	0.964323	NA	–	0.166385	0.569358	0.872	0.968

[a] Using decoder-based.
[b] Using stochastic ranking.
[c] Using over-penalty approach.
[d] Using penalty-based.
[e] Using Deb's method.

Table 5
Success rates of algorithms when compared with that of the ABC algorithm run through 240,000 objective function evaluation, dually. + indicates that algorithm is better while − indicates it is worse than other. If both algorithms show similar performance, they are both +.

Prb	ABC-HM		ABC-SR		ABC-ISR		ABC-OPA		ABC-ASCHEA		ABC-GA		ABC-SMES		ABC-PSO		ABC-DE	
	ABC	HM	ABC	SR	ABC	ISR	ABC	OPA	ABC	ASCHEA	ABC	GA	ABC	SMES	ABC	PSO	ABC	DE
g01	+	–	+	+	+	+	+	+	+	–	+	–	+	+	+	–	+	–
g02	–	+	+	–	+	–	+	–	+	–	+	–	+	–	+	–	+	–
g03	+	+	+	+	–	+	+	–	+	+	+	–	+	+	+	–	+	+
g04	+	–	+	+	+	+	+	+	+	–	+	–	+	+	+	+	+	+
g05	+	–	–	+	–	+	+	–	–	+	–	–	–	+	–	+	+	–
g06	+	–	+	–	–	+	–	+	+	–	+	–	+	–	–	+	+	–
g07	+	–	–	+	–	+	–	+	+	–	+	–	+	–	+	–	–	+
g08	+	–	+	+	+	+	+	+	+	+	+	–	+	+	+	+	+	+
g09	+	–	+	–	–	+	–	+	+	–	+	–	+	–	–	+	–	+
g10	+	–	+	–	–	+	–	+	–	+	+	–	+	–	–	+	–	+
g11	+	–	+	+	+	+	+	–	+	+	+	+	+	+	+	–	+	–
g12	+	+	+	+	+	+	+	–	+	–	+	+	+	+	+	–	+	+
g13	+	–	–	+	–	+	–	+	+	–	–	–	–	+	–	+	–	+
Total	12	3	10	9	6	12	8	8	11	5	11	2	11	8	8	7	9	8

problems and the value of ρ for g05 and g13 is 0.000%. Also, these problems have nonlinear equality constraints. g10 is linear and the value of ρ is 0.0020% for this problem. However, g10 has no linear equality and nonlinear equality constraints. Therefore, it is not possible to make any generalization for ABC algorithm such that it is better or not for a specific set of problems. In other words, it is not clear what characteristics of the test problems make it difficult for ABC. However, ABC algorithm produced comparable results to those obtained by the state-of-the-art algorithms used in the previous works although it does not employ any complicated mechanism introduced for constrained optimization and it is as simple as the version of the ABC used for unconstrained optimization [28].

It is a known fact that the performance of any evolutionary algorithm on constrained optimization is highly affected by the constraint handling method employed. In this work, since our aim was to evaluate the performance of the modified ABC algorithm on constrained optimization problems, we kept the modification to ABC algorithm minimum. The modification is associated mostly with selection process. Instead of a greedy selection used in the standard ABC algorithm for unconstrained problems, Deb's rules

Table 6
Statistical results obtained by ABC algorithm for 13 test functions over 30 independent runs using 350,000 objective function evaluations.

Problem	Optimal	Best	Mean	Worst	S.D.
g01	−15.000	−15.000	−15.000	−15.000	0.000
g02	0.803619	0.803611	0.795430	0.770319	0.009466
g03	1.000	1.000	1.000	1.000	0.000
g04	−30665.539	−30665.539	−30665.539	−30665.539	0.000
g05	5126.498	5126.487	5182.868	5374.430	68.584
g06	−6961.814	−6961.814	−6961.814	−6961.813	0.0004
g07	24.306	24.324	24.447	24.835	0.113
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.631	680.636	680.641	0.0026
g10	7049.25	7058.823	7220.106	7493.943	122.589
g11	0.75	0.750	0.750	0.750	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.757	0.975	1.000	0.051

Table 7
Success rates of algorithms when compared with that of the ABC algorithm run through 350,000 objective function evaluation, dually. + indicates that algorithm is better while − indicates it is worse than other. If both algorithms show similar performance, they are both +.

Prb	ABC-HM		ABC-SR		ABC-ISR		ABC-OPA		ABC-ASCHEA		ABC-GA		ABC-SMES		ABC-PSO		ABC-DE	
	ABC	HM	ABC	SR	ABC	ISR	ABC	OPA	ABC	ASCHEA	ABC	GA	ABC	SMES	ABC	PSO	ABC	DE
g01	+	−	+	+	+	+	+	+	+	−	+	−	+	+	+	−	+	−
g02	−	+	+	−	+	−	+	−	+	−	+	−	+	−	+	−	+	−
g03	+	+	+	+	−	+	+	−	+	+	+	−	+	+	+	−	+	+
g04	+	−	+	+	+	+	+	+	+	−	+	−	+	+	+	+	+	+
g05	+	−	−	+	−	+	+	−	−	+	−	−	−	+	−	+	+	−
g06	+	−	+	−	+	+	+	+	+	−	+	−	+	−	+	+	+	−
g07	+	−	−	+	−	+	−	+	+	−	+	−	+	−	+	−	−	+
g08	+	−	+	+	+	+	+	+	+	+	+	−	+	+	+	+	+	+
g09	+	−	+	−	−	+	−	+	+	−	+	−	+	−	−	+	−	+
g10	+	−	+	−	−	+	−	+	−	+	+	−	+	−	−	+	−	+
g11	+	−	+	+	+	+	+	−	+	+	+	+	+	+	+	−	+	−
g12	+	+	+	+	+	+	+	−	+	−	+	+	+	+	+	−	+	+
g13	+	−	−	+	−	+	−	+	+	−	−	−	−	+	−	+	−	+
Total	12	3	10	9	7	12	9	8	11	5	11	2	11	8	9	7	9	8

were employed in the version of ABC algorithm proposed for constrained problems. The performance of ABC algorithm might change depending on the constraint handling method used. Therefore, it is not fair to compare the performance of ABC algorithm with that of algorithms employing various constraint handling methods. It will be more reasonable to compare the performance of ABC with the algorithms using Deb's rules or using similar methods such as OPA. They can be considered similar since in Deb's rules and over-penalty approach, the feasibility always has a priority on the quality of the solution. Therefore, in order to evaluate the performance of ABC on constrained problems more fairly, ABC algorithm should be compared with GA, DE, PSO, SMES which use Deb's rules and OPA. When the comparison is made in this way, it is seen from Table 7 that in terms of success rates, ABC algorithm outperforms all algorithms employing Deb's rules and OPA.

In fact, the algorithms using Deb's rules lack diversity in the population because feasible solutions are preferred to infeasible ones. However, the diversity is one of the most important aspects to consider when designing a competitive constraint handling approach [29]. The diversity in the population is achieved by scout phase in ABC algorithm which directly allows infeasible solutions to be added to the population and onlooker phase that allows infeasible solutions to be selected probabilistically and inversely proportional to their constraint violation values.

The effect of control parameters of an algorithm on its performance is significant issue to be investigated since they dictate the search efficiency. However, while a parameter set produces optimal solutions on a particular landscape, the same set can result in failure for another one. Another issue is that an appropriate parameter set at the beginning may become inappropriate in next evaluations of the algorithm. Therefore, a parameter set is required to be optimal in general, to be robust, to produce good performance [30]. An analysis is desired to determine which parameter set yield the best performance. In order to find the most suitable values to be established for control parameters, analysis of the variance (ANOVA) based on the mean variance was used. ANOVA examines the statistically difference of one, two or more groups over one quantitative. The null hypothesis we established for ANOVA was that all population means are equal and the other hypothesis was that at least one mean is different from the others. Calculated F value is compared to the F value coming from the F distribution associated with the degrees of freedom and significance value. If this value is lower than 0.05, effect of the variable is considered to be statistically significant. Beside ANOVA, we conducted analysis of mean (ANOM) statistical test in order to decide the best parameter settings. Both ANOVA and ANOM statistics are calculated using MATLAB.

We examined 9 levels for control parameter *MR* from 0.1 to 0.9, 5 levels for *limit*$\{0.1 \times sn \times D,\ 0.5 \times sn \times D,\ sn \times D,\ 2 \times sn \times D,\ 4 \times sn \times D\}$ and 5 levels for *SPP*$\{0.1 \times sn \times D,\ 0.5 \times sn \times D,\ sn \times D,\ 2 \times sn \times D,\ 4 \times sn \times D\}$. We evaluated each control parameter in itself. Each level has 30 error values coming from each run.

In Tables 8–10, results of ANOVA test for *MR*, *Limit* and *SPP* control parameters are presented. In tables, SS indicates the sum of squares (SS) due to each source, DF indicates the degrees of freedom (df) associated with each source and MS indicates the mean Squares (MS), which is the ratio SS/df. F shows the F statistics, which is the ratio of the mean squares. Tables tell on which functions examined control parameter has significant effect on the error value. From the results in Table 8, g01, g06, g08 and g12 are not influenced by the values of *MR*. From the results given in Table 9, g01, g08 and g12 are also not influenced by the values of *Limit*. Control parameter *SPP* does not affect the error value coming from g01, g04, g08 and g12 functions.

Table 8
ANOVA table for the error obtained by 9 different *MR* values.

Function	SS	df	MS	F	$Prob > F$
g01	0	8	0	0	1
g02	0.00442	8	0.00055	8.24	0
g03	0.00042	8	5.28099e−005	199.49	0
g04	61136.1	8	7642.02	78.47	0
g05	758893.3	8	94861.7	2.82	0.0053
g06	0	8	0	−0	1
g07	29.8198	8	3.72748	57.08	0
g08	2.31112e−032	8	2.88889e−033	−3.77302e−016	1
g09	25.0808	8	8 3.1351	304.58	0
g10	1258569.7	8	157321.2	8.2	0
g11	6.20552e−007	8	7.7569e−008	4.53	0
g12	0	8	0	NaN	NaN
g13	3.62352	8	0.45294	28.47	0

Table 9
ANOVA table for the error obtained by 5 different *Limit* values.

Function	SS	df	MS	*F*	*Prob* > *F*
g01	4.73317e−028	4	1.18329e−028	3.77302e−016	1
g02	0.00018	4	0.00004	0.38	0.8253
g03	0.00039	4	9.78255e−005	206.33	0
g04	55877.1	4	13969.3	79.93	0
g05	219937.6	4	54984.4	2.89	0.0252
g06	617376.5	4	154344.1	140.85	0
g07	28.3687	4	7.09218	82.25	0
g08	0	4	0	−0	1
g09	22.5267	4	5.63167	377.56	0
g10	997158.2	4	249289.6	13.25	0
g11	4.78378e−007	4	4 1.19594e−007	6.5	0.0001
g12	0	4	0	NaN	NaN
g13	0.80775	4	0.20194	15.42	0

Table 10
ANOVA table for the error obtained by 5 different *SPP* values.

Function	SS	df	MS	*F*	*Prob* > *F*
g01	4.73317e−028	4	1.18329e−028	3.77302e−016	1
g02	0.00091	4	0.00023	2.89	0.0253
g03	5.41396e−007	4	1.35349e−007	187.09	0
g04	2.49742e−019	4	6.24356e−020	−7.91077e−014	1
g05	321282.5	4	80320.6	5.12	0.0008
g06	155.567	4	38.8917	46.84	0
g07	0.09214	4	0.02304	1.35	0.2575
g08	0	4	0	−0	1
g09	0.00066	4	0.00017	15.04	0
g10	182792.5	4	45698.1	3.69	0.0073
g11	1.68171e−008	4	4.20427e−009	4.56	0.0019
g12	0	4	0	NaN	NaN
g13	0.06132	4	0.01533	2.52	0.0449

Table 11
ANOM table for 9 different values of *MR*: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9.

Function	Above level	Below level limit	No significance
g01	–	–	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
g02	0.1, 0.9	0.3	0.2, 0.4, 0.5, 0.6, 0.7, 0.8
g03	0.1	0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	0.2
g04	0.1	0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	0.2
g05	0.9	–	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
g06	–	–	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
g07	0.1, 0.2	0.5, 0.6, 0.7, 0.8, 0.9	0.3, 0.4
g08	–	–	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
g09	0.1, 0.2	0.4, 0.5, 0.6, 0.7, 0.8, 0.9	0.3
g10	0.1	–	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
g11	0.1	–	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
g12	–	–	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
g13	0.5, 0.6, 0.7, 0.8, 0.9	0.1, 0.2, 0.3	0.4

Table 12
ANOM table for 5 different values of *Limit*: $0.1 \times sn \times D$, $0.5 \times sn \times D$, $sn \times D$, $2 \times sn \times D$, $4 \times sn \times D$.

Function	Above level	Below level limit	No significance
g01	–	–	0.1, 0.5, 1, 2, 4
g02	–	–	0.1, 0.5, 1, 2, 4
g03	0.1	0.5, 1, 2, 4	–
g04	0.1	0.5, 1, 2, 4	–
g05	0.1	0.5, 1, 2, 4	–
g06	0.1	0.5, 1, 2, 4	–
g07	0.1	0.5, 1, 2, 4	–
g08	–	–	0.1, 0.5, 1, 2, 4
g09	0.1	0.5	1, 2, 4
g10	0.1	–	0.5, 1, 2, 4
g11	–	–	0.1, 0.5, 1, 2, 4
g12	–	–	0.1, 0.5, 1, 2, 4
g13	1	0.1	0.5, 2, 4

Table 13
ANOM table for 5 different values of *SPP*: $0.1 \times sn \times D$, $0.5 \times sn \times D$, $sn \times D$, $2 \times sn \times D$, $4 \times sn \times D$.

Function	Above level	Below level limit	No significance
g01	–	–	0.1, 0.5, 1, 2, 4
g02	2	0.1	0.5, 1, 4
g03	0.1	0.5, 1, 2, 4	–
g04	0.1	–	0.5, 1, 2, 4
g05	4	–	0.1, 0.5, 1, 2
g06	0.1	0.5, 1, 2, 4	–
g07	–	–	0.1, 0.5, 1, 2, 4
g08	–	–	0.1, 0.5, 1, 2, 4
g09	0.1	1	0.5, 2, 4
g10	4	0.1	0.5, 1, 2
g11	0.1	–	0.5, 1, 2, 4
g12	–	–	0.1, 0.5, 1, 2, 4
g13	–	4	0.1, 0.5, 1, 2

In order to conclude which values for control parameters produce the best results, ANOM tables were produced. ANOM tables for *MR*, *Limit* and *SPP* are given in Tables 11–13, respectively. ANOM identifies any group that is performing differently from the rest by comparing the absolute deviations of group means from their overall mean. From the figures and tables, we found the appropriate ranges for control parameters that give the better or similar to group mean with respect to group average. Finally, from these ANOM tables by considering all functions, we can recommend the range [0.3–0.8] for the control parameter *MR*, the range $[0.5 \times sn \times D–sn \times D]$ for the control parameter *Limit* and the range $[0.1 \times sn \times D–2 \times sn \times D]$ for the control parameter *SPP*.

4. Conclusion

A modified version of ABC algorithm for constrained optimization problems has been introduced and its performance has been compared with that of the state-of-the-art algorithms. A statistical analysis of the parameters of the modified ABC algorithm has been conducted and appropriate values have been recommended. It has been concluded that ABC algorithm can be efficiently used for solving constrained optimization problems. The performance of ABC algorithm can be also tested for real engineering problems existing in the literature and compared with that of other algorithms. Also, the effect of constraint handling methods on the performance of ABC algorithm can be investigated in future works.

Appendix A.

$$\textbf{g01}: \text{Minimize} f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to

$$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0$$

$$g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0$$

$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0$$

$$g_4(\vec{x}) = -8x_1 + x_{10} \le 0$$

$$g_5(\vec{x}) = -8x_2 + x_{11} \le 0$$

$$g_6(\vec{x}) = -8x_3 + x_{12} \le 0$$

$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \le 0$$

$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \le 0$$

$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \le 0$$

where bounds are $0 \le x_i \le 1 (i = 1, \ldots, 9, 13)$, $0 \le x_i \le 100 (i = 10, 11, 12)$. The global optimum is at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, , 3, 3, 3, 1)$, $f(x^*) = -15$.

Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

$$\textbf{g02}: \text{Maximize} f(\vec{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

subject to

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \le 0$$

$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \le 0$$

where $n = 20$ and $0 \le x_i \le 10 (i = 1, \ldots, n)$. The known global maximum is at $x_i^* = 1/\sqrt{n} (i = 1, \ldots, n)$, $f(x^*) = 0.803619$. g_1 is close to being active ($g_1 = -10^{-8}$)

$$\textbf{g03}: \text{Maximize} f(\vec{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$

subject to

$$h(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

where $n = 10$ and $0 \le x_i \le 1 (1 = 1, \ldots, n)$. The global maximum is at $x_i^* = 1/\sqrt{(n)}\,(i = 1, \ldots, n)$ where $f(x^*) = 1$

$$\textbf{g04}: \text{Minimize} f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

subject to

$$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$$

$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0$$

$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2 - 110 \le 0$$

$$g_4(\vec{x}) = -80.51249 - 0.0071317x2x5 + 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0$$

$$g_5(\vec{x}) = 9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 - 25 \le 0$$

$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$$

where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$, $27 \le x_i \le 45\,(i = 3, 4, 5)$. The optimum solution is $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$, where $f(x^*) = -30665.539$. Constraints g_1 and g_6 are active.

g05 : Minimize $f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + \left(\frac{0.000002}{3}\right)x_2^3$

subject to

$$g_1(\vec{x}) = -x4 + x3 - 0.55 \le 0$$

$$g_2(\vec{x}) = -x3 + x4 - 0.55 \le 0$$

$$h_1(\vec{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_2(\vec{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_3(\vec{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

where $0 \le x_1 \le 1200$, $0 \le x_2 \le 1200$, $-0.55 \le x_3 \le 0.55$, and $-0.55 \le x_4 \le 0.55$. The best known solution is $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$, where $f(x^*) = 5126.4981$.

g06 : Minimize $f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$

subject to

$$g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$$

$$g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$$

where $13 \le x_1 \le 100$ and $0 \le x_2 \le 100$. The optimum solution is $x^* = (14.095, 0.84296)$ where $f(x^*) = -6961.81388$. Both constraints are active.

g07 : Minimize $f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$

subject to

$$g_1(\vec{x}) = -105 + 4x1 + 5x2 - 3x7 + 9x8 \le 0$$

$$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0$$

$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0$$

$$g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x4 - 120 \le 0$$

$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0$$

$$g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0$$

$$g_7(\vec{x}) = 0.5(x1 - 8)2 + 2(x2 - 4)2 + 3x_5^2 - x6 - 30 \le 0$$

$$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0$$

where $-10 \le x_i \le 10 (i = 1, \ldots, 10)$. The global optimum is $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$, where $f(x^*) = 24.3062091$. Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

g08 : Maximize $f(\vec{x}) = \dfrac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$

subject to

$$g_1(\vec{x}) = x_1^2 - x_2 + 1 \le 0$$

$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \le 0$$

where $0 \le x_i \le 10 (i = 1, 2)$. The optimum solution is located at $x^* = (1.2279713, 4.2453733)$, where $f(x^*) = 0.0095825$.

g09 : Minimize $f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$

subject to

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x3 + 4x_4^2 + 5x5 \le 0$$

$$g_2(\vec{x}) = -282 + 7x_2 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0$$

$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0$$

$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0$$

where $-10 \le x_i \le 10$, $(i = 1, \ldots, 7)$. The global optimum is $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$, where $f(x^*) = 680.6300573$. g_1 and g_4 constraints are active.

g10 : Minimize $f(\vec{x}) = x_1 + x_2 + x_3$

subject to

$$g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \le 0$$

$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \le 0$$

$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \le 0$$

$$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83.333333 \le 0$$

$$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \le 0$$

$$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \le 0$$

where $100 \le x_1 \le 10000$, $1000 \le x_i \le 10000, (i = 2, 3)$, $10 \le x_i \le 1000, (i = 4, \ldots, 8)$. The global optimum is $x^* = (579.19, 1360.13, 5109.92, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$, where $f(x^*) = 7049.25$. g_1, g_2 and g_3 are active.

g11 : Minimize $f(\vec{x}) = x_1^2 + (x_2 - 1)^2$

subject to

$$h(\vec{x}) = x_2 - x_1^2 = 0$$

where $-1 \le x_1 \le 1$, $-1 \le x_2 \le 1$. The optimum solution is $x^* = (\pm 1/\sqrt{(2)}, 1/2)$, where $f(x^*) = 0.75$.

g12 : Maximize $f(\vec{x}) = \dfrac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$

subject to

$$g_1(\vec{x}) = (x_i - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$$

where $0 \le x_i \le 10$, $(i = 1, 2, 3)$ and $p, r, q = 1, \ldots, 9$. The global optimum is located at $x^* = (5, 5, 5)$, where $f(x^*) = 1$.

g13 : Minimize $f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5}$

subject to

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 0$$

$$h_2(\vec{x}) = x_2 x_3 - 5x_4 x_5 = 0$$

$$h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$$

where $-2.3 \le x_i \le 2.3$ $(i = 1, 2)$, $-3.2 \le x_i \le 3.2$,$(i = 3, 4, 5)$. The global optimum is $x^* = (-1.717143, 1.5957091, -0.736413, -0.763645)$, where $f(x^*) = 0.0539498$.

Appendix B. Supplementary Data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.asoc.2010.12.001.

References

[1] K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization method for constrained optimization problems, in: Proceedings of the Euro-International Symposium on Computational Intelligence 2002, Press, 2002, pp. 214–220.

[2] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation 4 (1) (1995) 1–32.

[3] Z. Michalewicz, K. Deb, M. Schmidt, T. Stidsen, Evolutionary algorithms for engineering applications, in: K. Miettinen, P. Neittaanmäki, M.M. Mäkelä, J. Périaux (Eds.), Evolutionary Algorithms in Engineering and Computer Science, John Wiley and Sons, Chichester, England, 1999, pp. 73–94.

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

[5] Z. Michalewicz, C.Z. Janikow, Handling constraints in genetic algorithms, in: R.K. Belew, L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91), San Mateo, California, University of California, San Diego, Morgan Kaufmann Publishers, 1991, pp. 151–157.

[6] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation 4 (1) (1996) 1–32.

[7] J.C. Bean, A.B. Hadj-Alouane, A dual genetic algorithm for bounded integer programs, Technical Report TR 92–53, Department of Industrial and Operations Engineering, The University of Michigan, 1992, to appear in R.A.I.R.O.-R.O. (invited submission to special issue on GAs and OR).

[8] A. Homaifar, S.H.Y. Lai, X. Qi, Constrained optimization via genetic algorithms, Simulation 62 (4) (1994) 242–254.

[9] J. Joines, C. Houck, On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, in: D. Fogel (Ed.), Proceedings of the first IEEE Conference on Evolutionary Computation, IEEE Press, Orlando, Florida, 1994, pp. 579–584.

[10] Z. Michalewicz, N.F. Attia, Evolutionary optimization of constrained problems, in: Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific, 1994, pp. 98–108.

[11] J.T. Richardson, M.R. Palmer, G. Liepins, M. Hilliard, Some guidelines for genetic algorithms with penalty functions, in: J.D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89), George Mason University, Morgan Kaufmann Publishers, San Mateo, California, June, 1989, pp. 191–197.

[12] M. Schoenauer, S. Xanthakis, Constrained GA optimization, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93), University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, San Mateo, California, July, 1993, pp. 573–580.

[13] D. Powell, M.M. Skolnick, Using genetic algorithms in engineering design optimization with non-linear constraints, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93), University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers, San Mateo, California, July, 1993, pp. 424–431.

[14] Z. Michalewicz, G. Nazhiyath, Genocop III: a co-evolutionary algorithm for numerical optimization with nonlinear constraints, in: D.B. Fogel (Ed.), Proceedings of the Second IEEE International Conference on Evolutionary Computation, IEEE Press, Piscataway, NJ, 1995, pp. 647–651.

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

[16] J. Paredis, Co-evolutionary constraint satisfaction, in: Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, Springer-Verlag, New York, 1994, pp. 46–55.

[17] I.C. Parmee, G. Purchase, The development of a directed genetic search technique for heavily constrained design spaces, in: I.C. Parmee (Ed.), Adaptive Computing in Engineering Design and Control-'94, University of Plymouth, Plymouth, UK, 1994, pp. 97–102.

[18] H. Myung, J.-H. Kim, D.B. Fogel, Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems, in: J.R. McDonnell, R.G. Reynolds, D.B. Fogel (Eds.), Proceedings of the Fourth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 449–463.

[19] R.G. Reynolds, Z. Michalewicz, M. Cavaretta, Using cultural algorithms for constraint handling in GENOCOP, in: J.R. McDonnell, R.G. Reynolds, D.B. Fogel (Eds.), Proceedings of the Fourth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 298–305.

[20] S.B. Hamida, M. Schoenauer, ASCHEA: new results using adaptive segregational constraint handling, in: Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002), vol. 1, IEEE Service Center, Piscataway, NJ, May, 2002, pp. 884–889.

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

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

[23] E. Mezura-Montes, C.A. Coello Coello, A simple multimembered evolution strategy to solve constrained optimization problems. Technical Report EVOCINV-04–2003, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D.F., México, 2003. Available in the Constraint Handling Techniques in Evolutionary Algorithms Repository at http://www.cs.cinvestav.mx/~constraint/.

[24] D. Karaboga, An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[25] V. Tereshko, Reaction–diffusion model of a honeybee colony's foraging behaviour, in: M. Schoenauer (Ed.), Parallel Problem Solving from Nature VI, vol. 1917 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2000, pp. 807–816.

[26] A. Muñoz-Zavala, A.A. Hernández, E.R.V. Diharce, Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), in: H.-G. Beyer, U.-M. O'Reilly, D.V. Arnold, W. Banzhaf, C. Blum, E.W. Bonabeau, E. Cant Paz, D. Dasgupta, K. Deb, J.A. Foster, E.D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G.R. Raidl, T. Soule, A. Tyrrell, J.-P. Watson, E. Zitzler (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005), vol. 1, New York, June, 2005, ACM Press, Washington DC, USA, pp. 209–216, ISBN 1–59593-010–8.

[27] D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, K.V. Price (Eds.), New Ideas in Optimization. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[28] B. Basturk, D. Karaboga, An artificial bee colony (abc) algorithm for numeric function optimization, in: IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA, May, 2006.

[29] E. Mezura-Montes, C.A. Coello Coello, Adding a diversity mechanism to a simple evolution strategy to solve constrained optimization problems, in: The 2003 Congress on Evolutionary Computation, 2003 (CEC'03), vol. 1, December, 2003, pp. 6–13.

[30] K. DeJong, Parameter setting in eas: a 30 year perspective, in: Parameter Setting in Evolutionary Algorithms, 2007, pp. 1–18.