J Intell Manuf (2012) 23:1001–1014
DOI 10.1007/s10845-010-0393-4

Artificial bee colony algorithm for large-scale problems and engineering design optimization

Bahriye Akay · Dervis Karaboga

Received: 16 July 2009 / Accepted: 18 February 2010 / Published online: 31 March 2010

Abstract Engineering design problems are generally large scale or nonlinear or constrained optimization problems. The Artificial Bee Colony (ABC) algorithm is a successful tool for optimizing unconstrained problems. In this work, the ABC algorithm is used to solve large scale optimization problems, and it is applied to engineering design problems by extending the basic ABC algorithm simply by adding a constraint handling technique into the selection step of the ABC algorithm in order to prefer the feasible regions of entire search space. Nine well-known large scale unconstrained test problems and five well-known constrained engineering problems are solved by using the ABC algorithm and the performance of ABC algorithm is compared against those of state-of-the-art algorithms.

Keywords Unconstrained optimization · Constrained optimization · Mechanical design problems · Artificial bee colony

Introduction

Evolutionary algorithms (EAs) are known as general-purpose optimization algorithms which are capable of finding near-optimal solutions to the numerical, real-valued test problems for which exact and analytical methods do not produce optimal solutions within a reasonable computation time. One of the evolutionary algorithms which has been introduced recently is the Differential Evolution (DE) algorithm (Storn and Price 1995). The DE algorithm was proposed by Storn and Price in 1995 especially for to solve numerical problems. In recent years, swarm intelligence has become a research interest to many research scientists in the related fields. The term "swarm" is used in a general manner to refer to any restrained collection of interacting agents or individuals. The classical example of a swarm is bees' swarming around their hive, but the metaphor can easily be extended to other systems with similar architecture. An ant colony can be thought of as a swarm whose individual agents are ants; a flock of birds is a swarm of birds. The particle swarm optimization (PSO) algorithm, simulating the behaviour of a bird swarm, was introduced by (Kennedy and Eberhart 1995). PSO is a population based stochastic optimization technique and well adapted to the optimization of nonlinear functions in multidimensional space.

Although modern heuristic algorithms were first proposed for unconstrained optimization problems, they have been extended to solve engineering design problems (Mezura-Montes and Coello Coello 2005b; Ray and Liew 2003; He et al. 2004; Parsopoulos and Vrahatis 2005). Mezura-Montes and Coello Coello (2005b) proposed an evolutionary-based approach to solve engineering design problems without using penalty functions. The approach generates more solutions inside the feasible region and near the boundaries by using genetic operators to sample the feasible region well enough to reach better feasible solutions. Ray and Liew (2003) introduced a meta-heuristic called the Society and civilization algorithm (SCA) that utilizes the intra- and inter-society interactions within a formal society; the civilization model is used to solve single objective constrained optimization problems. The basic idea of the SCA technique is that social interactions enable individuals to adapt and improve faster than biological evolution based on genetic inheritance alone. In order to handle constraints, the SCA employs a non-dom-

B. Akay (✉) · D. Karaboga
Department of Computer Engineering, Erciyes University, 38039 Melikgazi, Kayseri, Turkiye
e-mail: bahriye@erciyes.edu.tr

D. Karaboga
e-mail: karaboga@erciyes.edu.tr

inance scheme that eliminates the problem of scaling and aggregation that is common among penalty-function-based methods. He et al. (2004) presented an improved PSO to solve engineering mechanical design optimization problems by using a constraint handling method called the 'fly-back mechanism' which maintains a feasible population. Parsopoulos and Vrahatis (2005) applied the Unified Particle Swarm Optimization method to constrained engineering optimization problems. For this purpose, they employed a penalty function approach, and the algorithm is modified to preserve the feasibility of the encountered solutions.

For optimizing numerical multivariable functions, Karaboga (2005) has described an Artificial Bee Colony (ABC) algorithm which mimics the foraging behaviour of a honey bee swarm. In the field of evolutionary computation, it is common to compare different algorithms using a large test set, especially when the test involves function optimization (Boyer et al. 2005). Real-world problems usually have many design parameters that should be considered in the design process. Algorithms that are not robust to large-scale problems cannot preserve their effectiveness against high dimensionality. In this study, the Artificial Bee Colony algorithm, which has been introduced recently for unconstrained optimization problems, is used for optimizing large-scale unconstrained optimization problems and engineering design problems. Its performance is compared to that of other state-of-the-art algorithms mentioned above. The rest of the paper is organized as follows: In Sect. "Constraint handling methods", constraint handling techniques are articulated. In Sect. "Artificial Bee Colony algorithm", Artificial Bee Colony algorithm and its adaptation to constrained optimization are presented. Settings and experimental results on unconstrained large-scale benchmark problems are given in Sect. "Experiments 1: large-scale unconstrained optimization" and those on constrained mechanical design problems are presented in Sect. "Experiments 2: constrained mechanical design optimization".

Constraint handling methods

An engineering design problem which is usually a constrained optimization (CO) problem involves problem-specific constraints. A general constrained optimization problem is to find **x** so as to

$$\begin{aligned}
&\text{minimize} \quad f(\mathbf{x}), \quad \mathbf{x} = (x_1, x_2, \ldots, x_i, \ldots x_n)\\
&\text{subject to :}\\
&g_j(\mathbf{x}) \le 0, \quad \text{for } j = 1, \ldots, q \text{ (inequality constraint)}\\
&h_j(\mathbf{x}) = 0, \quad \text{for } j = q+1, \ldots, m \text{ (equality constraint)}
\end{aligned}$$

where **x** represents a solution to the design problem, x_i is called the design parameter, $f(\mathbf{x})$ is the objective function to be minimized and m is the number of constraints.

The presence of constraints in engineering design problems requires some additional constraint handling approaches which need to be incorporated into optimization algorithms. These approaches direct the search towards the region containing acceptable solutions called the feasible region (Zhang et al. 2006).

Complete search algorithms search all possible assignments of values to variables in the search space, but they have the disadvantage of being time consuming. Incomplete search methods search the space either non-systematically or in a systematical manner (Bartak et al. 2008). Incomplete search methods dealing with the constraints were grouped into four categories by Koziel and Michalewicz (1999):

- Methods based on preserving feasibility of solutions (Michalewicz and Janikow 1991);
- Methods based on penalty functions (Bean and Hadj-Alouane 1992; Homaifar et al. 1994; Joines and Houck 1994; Michalewicz and Attia 1994);
- Methods that make a clear distinction between feasible and infeasible solutions (Richardson et al. 1989; Schoenauer and Xanthakis 1993; Powell and Skolnick 1993; Michalewicz and Nazhiyath 1995; Deb 2000);
- Other hybrid methods (Paredis 1994; Parmee and Purchase 1994; Myung et al. 1995; Reynolds et al. 1995)

Methods based on preserving feasibility of solutions use operators which transform infeasible individuals into feasible individuals. This method assumes linear constraints only and a feasible starting point (or a feasible initial population) (Michalewicz and Janikow 1991). In the methods based on penalty functions, a constrained problem is solved as an unconstrained one, where the objective function is designed such that infeasible solutions are characterized by high function values (in minimization cases). Although the use of penalty functions is very common since its simplicity and direct applicability (Smith and Coit 1997; Coello Coello 1999; Yeniay 2005; Parsopoulos and Vrahatis 2002), they have several drawbacks, too. For example they require a careful fine tuning of the penalty factors that estimate the degree of penalization to be applied (Smith and Coit 1997; Mezura-Montes and Coello Coello 2005a; Coello Coello 2002). In the category including methods based on a search for feasible solutions, there are a few methods which emphasize the distinction between feasible and infeasible solutions in the search space. One method considers the problem constraints in a sequence switching from one constraint to another until the number of feasible individuals in the population meets a sufficient number. The second method is based on an assumption that any feasible solution is better than any infeasible one (Richardson et al. 1989; Deb 2000). The third method repairs infeasible individuals (Koziel and Michalewicz 1999). The last category includes hybrid methods that combine evolutionary computation techniques with deterministic procedures for numerical optimization procedures (Koziel and Michalewicz 1999).

Artificial Bee Colony algorithm

The Artificial Bee Colony algorithm proposed by Karaboga (2005) simulates the foraging behaviour of a bee colony and it was successfully applied to optimize numerical problems (Karaboga 2005; Basturk and Karaboga 2006; Karaboga and Basturk 2007b, 2008; Karaboga and Akay 2007; Karaboga et al. 2007; Karaboga and Basturk 2007a; Karaboga and Akay 2009). A colony of bees tries to find rich food sources in order to maximize the amount of nectar unloaded at the hive. Although there are several task classifications of bees in a real hive, Artificial Bee Colony uses minimal model that mimics the foraging behaviour of bees comprising of employed bees, onlooker bees and scouts. An employed bee finds a food source position by modifying the position in her memory and evaluates the nectar amount of each new source and memorizes the better one (greedy selection). Employed bees share information related with the quality of the food source they are exploiting, on the dance area. Onlooker bees find food sources based upon the information coming from employed bees. More profitable sources are more likely to be chosen by onlookers. An onlooker bee chooses a food source depending on this information (probabilistic selection) and produces a modification on this source. In order to determine the better source, a greedy selection is applied. In the nature of real bees, if a food source is not worth exploiting anymore, it is abandoned by the bees, and the employed bee of that source becomes a scout searching the environment randomly or by an internal motivation. The main steps of ABC algorithm are given on a flowchart seen in Fig. 1.

In order to solve constrained optimization problems, since the drawbacks of the penalty-based methods and the methods starting with feasible solutions, we incorporated Deb's rules (Deb 2000) into the Artificial Bee Colony algorithm, introduced in (Karaboga 2005), in order to prefer feasible solutions to infeasible ones or a low-cost infeasible solution to an expensive infeasible solution. Therefore, the algorithm promises to search in the feasible regions.

Since initialization with feasible solutions is a very time consuming process and in some cases it is impossible to produce a feasible solution randomly, the ABC algorithm does not consider the initial population to be feasible. Besides, the ABC algorithm does not employ a penalty function that requires a careful fine tuning of the penalty factors and penalty function itself. The structure of the algorithm already directs the solutions to the feasible region by means of Deb's three heuristic rules (Deb 2000) which are used instead of greedy selection. Deb's method uses a tournament selection operator, where two solutions are compared based on the following criteria:

1. Any feasible solution is preferred to any infeasible solution,
2. Among two feasible solutions, the one having a better objective function value is preferred,
3. Among two infeasible solutions, the one having the smaller constraint violation is preferred.

The Pseudo-code of the ABC algorithm proposed to solve constrained problems is given below:

```
1: Initialize the population of solutions $x_{i,j}$, $i = 1 \dots SN$, $j = 1 \dots D$
2: Evaluate the population
3: cycle = 1
4: **repeat**
5:   Produce a new solution $\boldsymbol{\upsilon}_i$ for each employed bee by using (1) and evaluate it
     $$\upsilon_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), & \text{if } R_j < MR \\ x_{ij}, & \text{otherwise} \end{cases} \quad (1)$$
     where $\phi_{ij}$ is a random number in the range [−1,1], $k \in \{1, 2, \dots SN\}$ ($SN$: number of solutions in the colony) is randomly chosen index. Although $k$ is determined randomly, it has to be different from $i$. $R_j$ is a randomly chosen real number in the range [0,1] and $j \in \{1, 2, \dots D\}$ ($D$: dimension of the problem). $MR$, modification rate, is a control parameter.
6:   Apply a selection process between $\boldsymbol{\upsilon}_i$ and $\mathbf{x}_i$ based on Deb's method
7:   Calculate the probability values $p_i$ for the solutions using fitness of the solutions and the constraint violations ($CV$) by (2)
     $$p_i = \begin{cases} 0.5 + \left( \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \right) * 0.5 & \text{if solution is feasible} \\ \left( 1 - \frac{CV}{\sum_{i=1}^{SN} CV} \right) * 0.5 & \text{if solution is infeasible} \end{cases} \quad (2)$$
     $CV$ is defined by (3)
     $$CV = \sum_{j=1, \text{if } g_j(\mathbf{x})>0}^{q} g_j(\mathbf{x}) + \sum_{j=q+1}^{m} h_j(\mathbf{x}) \quad (3)$$
8:   For each onlooker bee, produce a new solution $\boldsymbol{\upsilon}_i$ by (1) in the neighbourhood of the solution selected depending on $p_i$ and evaluate it
9:   Apply selection process between $\boldsymbol{\upsilon}_i$ and $\mathbf{x}_i$ based on Deb's method
10:  If Scout Production Period ($SPP$) is completed, determine the abandoned solutions by using "limit" parameter for the scout, if it exists, replace it with a new randomly produced solution by (4)
     $$\mathrm{x}_\mathrm{i}^\mathrm{j} = x_{\min}^j + rand(0,1)(x_{\max}^j - x_{\min}^j) \quad (4)$$
```

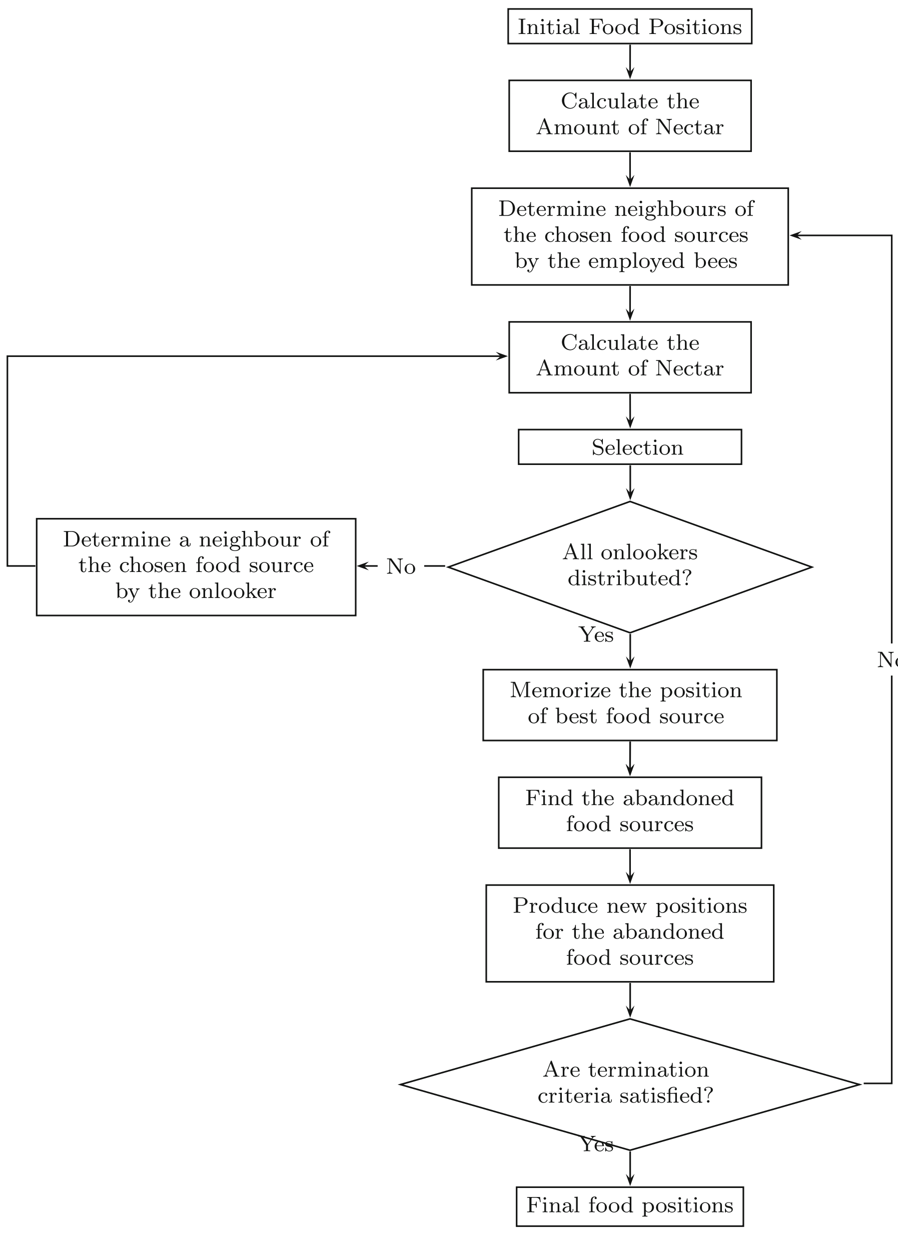
Initial Food Positions
Calculate the Amount of Nectar
Determine neighbours of the chosen food sources by the employed bees
Calculate the Amount of Nectar
Selection
All onlookers distributed?
No
Determine a neighbour of the chosen food source by the onlooker
Yes
Memorize the position of best food source
Find the abandoned food sources
Produce new positions for the abandoned food sources
Are termination criteria satisfied?
No
Yes
Final food positions

Fig. 1 Flowchart of the Artificial Bee Colony algorithm

```
11:     Memorize the best solution achieved so far
12:     cycle = cycle+1
13: until cycle = MCN
```

The scout production process of the algorithm provides a diversity mechanism that allows new and probably infeasible individuals to be inserted into the population. In the version of the ABC algorithm proposed for constrained optimization problems, artificial scouts are produced at a predetermined period of cycles (SPP) for discovering new food sources randomly. At each SPP cycle, it is controlled if there is an abandoned food source or not. If there is, the scout production process is carried out.

The probability calculation is different from the one employed in the version for unconstrained optimization problems. Since the ABC algorithm does not need to generate feasible solutions in initialization, and infeasible solutions are allowed to be added to the colony via the scout unit, a probability value is assigned for each infeasible solution in the colony as for in a feasible solution. But the ABC algorithm does feasible solutions credit by raising their calculated the probability values depending on the proportion of their fitness values out of feasible solutions fitness values sum. The probability value used by onlookers to choose a food source site is calculated for an infeasible solution depending on the proportion of its constraint violation out of the sum of all constraints violations.

Experiments 1: large-scale unconstrained optimization

Benchmark problems that we used in order to test the performance of the algorithms are given in Table 1. Sphere function is a convex, unimodal function which has no local minimum except the global one. Step has one minimum, and it is a discontinuous function which represents the problem of flat surfaces. Flat surfaces are obstacles for optimization algorithms which do not have variable step sizes, because they do not give any information as to which direction is favourable (Digalakis and Margaritis 2002). The surface of Schwefel function is composed of a great number of peaks and valleys. The function has a second best minimum far from the global minimum where many search algorithms are trapped. Main difficulty of Rosenbrock function is that it has a nonlinear deep valley with the shape of a parabola that leads to the global minimum and nonlinear interaction among the variables. We used the extended version to n variable of this function in our experiments. Rastrigin function was constructed from Sphere adding a modulator term. Its contour is made up of a large number of local minima whose value increases with the distance to the global minimum. Ackley has an exponential term that covers its surface with numerous local minima. In order to obtain good results for this function, the search strategy must combine the exploratory and exploitative components efficiently. Griewank function has a product term that introduces interdependence among the variables on which an algorithm optimizing each variable independently fails. As in Ackley function, the optima of Griewank function are regularly distributed. Penalized functions are difficult due to the combinations of different periods of the sine function (Digalakis and Margaritis 2002; Boyer et al. 2005).

Results of the ABC algorithm were taken for $D = 10$, 100 and 500 and were compared to those of the DE and PSO algorithms. In all experiments, common parameters such as population number, maximum evaluation number were chosen the same for all algorithms. Population size was 100, and the maximum evaluation number was 1.000.000 for all functions. Each of the experiments was repeated 30 times with different random seeds. The other specific parameters of algorithms are given below:

PSO settings: Cognitive and social components are constants that can be used to change the weighting between personal and population experience, respectively. In our experiments, cognitive and social components were both set

Table 1 Benchmark problems used in experiments 1

	Function	Range	Formulation
1	Sphere	$[-100, 100]^n$	$f(x) = \sum_{i=1}^{n} x_i^2$
2	Step	$[-100, 100]^n$	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
3	Schwefel	$[-500, 500]^n$	$f(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$
4	Rosenbrock	$[-100, 100]^n$	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
5	Dixon-Price	$[-10, 10]^n$	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
6	Rastrigin	$[-5.12, 5.12]^n$	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
7	Griewank	$[-600, 600]^n$	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
8	Ackley	$[-32, 32]^n$	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
9	Penalized	$[-50, 50]^n$	$f(x) = \frac{\pi}{n}\left\{\begin{array}{l} 10\sin^2(\pi y_1) \\ + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] \\ +(y_n - 1)^2 \end{array}\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$

Table 2 Results of PSO, DE and ABC algorithms on unconstrained large-scale benchmark problems

Functions	*D* = *10*			D = 100			D = 500		
	PSO	DE	ABC	PSO	DE	ABC	PSO	DE	ABC
Sphere	0	0	0	0	0	0	181.160	20.330	8.71E-7
Step	0	0	0	1.700	0	0	1621.000	1998.030	0
Schwefel	−2654.030	−4177.990	−4189.830	−20100.400	−31182.500	−41898.300	−98168.070	−138152.030	−190906.660
Rosenbrock	0.4265	0	0.013	113.144	132.349	0.055	10.90E+05	87.17E+09	1007.870
Dixon-price	0.666	0.666	0	2.076	0.666	1.26E-06	15315.110	2636.320	309.600
Rastrigin	7.363	0	0	148.249	133.114	0	1033.040	594.690	87.960
Griewank	0.059	0.004	0	0.049	0.001	0	2.200	0.645	0
Ackley	0	0	0	0.732	0	0	3.690	13.000	0.058
Penalized	0	0	0	0.185	0.013	0	5.290	14.81E + 09	3.46E-8

to 1.8. Inertia weight, which determines how the previous velocity of the particle influences the velocity in the next iteration, was 0.6.

DE settings: In DE, a trial solution for a parent is produced after taking the recombination of a mutant vector and the parent vector (solution). F is a real constant which affects the differential variation between two solutions and was set to 0.5 in our experiments. Value of the crossover rate, which controls the change of the diversity of the population, was 0.9 (Corne et al. 1999).

ABC settings: Except common (population number and maximum evaluation number), only one control parameter considered in ABC is “limit”. A food source will not be exploited and is assumed to be abandoned in case the “limit” is exceeded. This means that the solution that exceeds the limit value will not be improved any further. We defined a relation between the limit value and the dimension of the problem and the colony functions size, and used $SN * D * 0.5$ to determine the value of limit.

Mean best values produced by the algorithms are presented in Table 2. In order to make the comparison clearer, values below E-12 were assumed to be 0. As seen from Table 2, in case of dimension is 10, the ABC algorithm finds the global optimum values on Sphere, Step, Schwefel, Dixon-Price, Rastrigin, Griewank, Ackley and Penalized functions (eight functions) while DE finds on Sphere, Step, Rosenbrock, Rastrigin, Ackley and Penalized functions (six functions) and PSO finds on Sphere, Step, Ackley and Penalized functions (four functions). From these results, ABC outperforms DE on Schwefel, Dixon-Price and Griewank functions and DE outperforms ABC just on Rosenbrock function. While ABC outperforms PSO on Schwefel, Rosenbrock, Dixon-Price, Rastrigin and Griewank functions, PSO cannot outperform ABC on any function. When the dimension is 100, ABC shows better performance than both DE and PSO on all of the functions except Ackley function. On Ackley function, results of the DE and ABC algorithms are similar. Although DE performs better than ABC on Rosenbrock when $D = 10$, by the increase in dimension, the performance of DE deteriorates, but ABC preserves its robustness. When the dimension is incremented to 500, the ABC algorithm again produces the best results. From the results, it can be said that, as the number of function variables increases, the performance of the ABC algorithm stands out much more.

Since the benchmark set contains uni-modal (Sphere, Rosenbrock, Step), multi-modal (Schwefel, Dixon-Price, Rastrigin, Griewank, Ackley, Penalized), separable (Sphere, Step, Rastrigin), non-separable (Schwefel, Rosenbrock, Dixon-Price, Griewank, Ackley, Penalized), non-symmetric (Dixon-Price) problems, the superiority of the ABC algorithm can be explained by its structure which combines explorative and exploitative processes in a balanced manner. In the PSO algorithm, the particles fly in the search space by the guidance of the particle’s own best and the swarm’s best while the ABC algorithm does not use the guidance of the best solution. Although guidance of the best solution may improve the local convergence speed of the algorithm, it may attenuate the exploration capability of the algorithm if the algorithm does not have a suitable global search operator. ABC algorithm has efficient global search mechanisms conducted by a probabilistic selection scheme in its onlooker bees phase and a random selection scheme in its scout phase. Moreover, the PSO algorithm does not have an operator providing diversity in the population as ABC does in the scout phase. Although, the DE and ABC algorithms both have mutation operators weighting the difference of solutions, the DE algorithm uses a constant scaling factor while the ABC algorithm uses a random number in the range $[-1, 1]$. The DE algorithm has a crossover operator which makes it converge faster in initial generations than ABC but which may result in premature convergence in case of multi-modal problems since the DE algorithm does not have an operator maintaining sufficient diversity.

Experiments 2: constrained mechanical design optimization

In the second part of the experiments, we employed five well-known constrained engineering problems (Rao 1996): Welded Beam, Pressure Vessel, Tension/ Compression Spring, Speed Reducer, Gear Train (See Appendix A).

Welded beam design minimizes the cost of the beam subject to constraints on shear stress, τ , bending stress in the beam, σ , buckling load on the bar, P_c , end deflection of the beam, δ, and side constraints. This problem consists of a nonlinear objective function, five nonlinear and two linear inequality constraints. The solution is located on the boundaries of the feasible region. The ratio of feasible region to entire search space is quite small for welded beam problem. Techniques using adaptive ϵ for handling constraints can be effective for solving this problem.

The pressure vessel problem is to minimize the total cost of material, forming and welding of a cylindrical vessel. Pressure vessel problem has a nonlinear objective function, a nonlinear and three linear inequality constraints. Since the problem has two discrete variables and two continuous variables, it is a mixed discrete-continuous constrained optimization problem.

The tension/compression problem deals with minimizing of the weight of the tension/compression spring subject to constraints on the minimum deflection, shear stress, surge frequency, diameter and design variables. This problem has a nonlinear objective function, a linear and three nonlinear inequality constraints.

The aim of the speed reducer design is to minimize the weights of the speed reducer subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts. Speed reducer problem has seven nonlinear and four linear constraints. Four constraints are active at the best known feasible solution [3.5000 0.7000 17.0000 7.3000 7.7153 3.3502 5.2867] producing a 2994.34 kg gearbox.

Gear train design aims to minimize the cost of the gear ratio of the gear train. It has only boundary constraints in the parameters. Variables to be optimized are in discrete form since each gear has to have an integral number of teeth. Handling with discrete variables may increase the complexity of the problem.

The number of linear and nonlinear inequality constraints of the problems is given in Table 3. 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 (Michalewicz et al. 1999).

Details of the engineering design problems we used in the experiments are given in Appendix A.

Table 3 Number of linear and nonlinear inequality constraints of the problems

Problem	LI	NI
Welded beam	2	5
Pressure vessel	3	1
Tension/comp. spring	1	3
Speed reducer	4	7
Gear train	0	0

LI The number of linear inequalities, *NI* The number of nonlinear inequalities

In the experiments, results of the SCA (Ray and Liew 2003), versions of the Particle Swarm Optimization algorithm (PSO, UPSOm) (He et al. 2004; Parsopoulos and Vrahatis 2005), the Evolution Strategy ($\mu + \lambda - ES$) (Mezura-Montes and Coello Coello 2005b) and the ABC algorithms are compared to each other.

The values of the algorithm-specific control parameters are given in Table 4. Discrete variables were handled by truncating the real value to its closest integer value. We conducted 30 independent experiments for each problem. The best and mean values, standard deviations and the maximum evaluation numbers in order to terminate the algorithms are reported in Table 5. While the best solution, indicated as “Best” in the table, shows the ability of an algorithm to find the optimal, the mean and standard deviation values give information about the robustness of the algorithm. The maximum number of evaluations may be a metric of convergence rate.

Depending on the results in Table 5, on welded beam problem, in which the solution is located on the boundaries of the feasible region, the best solutions are produced by $(\mu+\lambda)$-ES and ABC algorithms. When the optimum is located on the boundaries of the feasible space, the problem gets more difficult and the solution has neighbours that are both feasible and infeasible. Although the best solutions found by $(\mu + \lambda)$-ES and ABC algorithms are the same, the mean values of 30 runs show that the ABC algorithm is more successful in solving the problem for overall runs. The evolution of the mean best result obtained by the ABC algorithm through 30 runs for Welded Beam problem is plotted on Fig. 2a.

For pressure vessel problem, $(\mu + \lambda)$-ES is the most successful algorithm in terms of “best” solution. The ABC algorithm and the PSO of He et al. (2004) produce just about the same best solutions, although the mean of solutions obtained by ABC algorithm is the lowest in all of the algorithms. Convergence of the mean best result produced by ABC algorithm through 30 runs for Pressure Vessel problem for the first 500 cycles is figured out on Fig. 2b.

For tension/compression spring problem, in terms of the best solution the PSO and ABC algorithms deliver equal performances and the mean results of the PSO and ABC algorithms are relatively close. However, the maximum evalua-

Table 4 The values of the control parameters of the algorithms

SCA		PSO		$(\mu + \lambda)$-ES		UPSOm		ABC	
CS	10**D*	SS	30	μ	15	χ	0.729	CSabc	30
TS	1,000	MGN	1,000	λ	100	c_1	2.05	MCN	1,000
σ	0.25~1	ω	0.8	S_r	0.97	c_2	2.05	MR	0.9
		c_1	0.5	MGN	300	NR	1	SPP	400
		c_2	0.5	LR	$\tau = \left(\sqrt{2\sqrt{n}}\right)^{-1}$ $\tau' = \left(\sqrt{2n}\right)^{-1}$	UF	0.1	Limit	CSabc**D**5
				MSS	$\sigma_i(0) = 0.4\left(\Delta xi/\sqrt{n}\right)$	MGN	5,000		
						SS	20		
						MtR	0.01		
						w1	100		
						w2	100		

D Dimension of the problem, *CS* Civilization size, *TS* Time step, σ variance of normal distribution, *SS* Swarm size, *MGN* Maximum generation number, ω inertia, c_1 cognitive component, c_2 social component, S_r Selection ratio, *LR* Learning rate, *MSS* Mutation step size, χ Constriction factor, *NR* Neighbourhood radius, *UF* Unification factor, *MtR* Mutation rate, *CSabc* Colony size, *MCN* Maximum cycle number, *SPP* Scout production period, *MR* Modification rate

Table 5 Statistical results of the SCA (Ray and Liew 2003)), PSO (He et al. 2004), $(\mu + \lambda)$-ES (Mezura-Montes and Coello Coello 2005b), UPSOm (Parsopoulos and Vrahatis 2005) and ABC algorithms

Problem	Stats.	SCA[a]	PSO[a]	$(\mu + \lambda)$-ES	UPSOm	ABC
Welded beam	Best	NA	NA	**1.724852**	1.92199	**1.724852**
	Mean	NA	NA	1.777692	2.83721	**1.741913**
	St. Dev	NA	NA	8.8E-2	0.682980	**3.1E-02**
	Evaluations	NA	NA	30,000	100,000	30,000
Pressure vessel	Best	6171.00	6059.7143	**6059.701610**	6544.27	6059.714736
	Mean	6335.05	6289.92881	6379.938037	9032.55	**6245.308144**
	St. Dev	NA	3.1E+2	2.1E+2	995.573	**2.05.E+02**
	Evaluations	20,000	30,000	30,000	100,000	30,000
Ten/comp.spring	Best	0.012669	**0.012665**	0.012689	0.0131200	**0.012665**
	Mean	0.012923	**0.012702**	0.013165	0.0229478	0.012709
	St. Dev	5.9E-4	**4.1E-5**	3.9E-4	0.00720571	0.012813
	Evaluations	25,167	**15000**	30,000	100,000	30,000
Speed reducer	Best	**2994.744241**	NA	2996.348094	NA	2997.058412
	Mean	3001.758264	NA	**2996.348094**	NA	2997.058412
	St. Dev	4.0E+0	NA	**0**	NA	0
	Evaluations	54,456	NA	30,000	NA	30,000
Gear train	Best	NA	NA	NA	**2.70085E-12**	**2.700857E-12**
	Mean	NA	NA	NA	**3.80562 E-8**	**3.641339E-10**
	St. Dev	NA	NA	NA	**1.09631 E-7**	**5.525811E-10**
	Evaluations	NA	NA	NA	100,000	**60**

Bold face indicates the winner of the algorithms, NA: not available

[a] The welded beam problems are different from the one employed in this paper

tion number of PSO is smaller than that of the ABC algorithm. The evolution of mean best result in the first 500 cycles obtained by ABC algorithm is given on Fig. 2c.

For speed reducer problem, the best result is found by SCA, although the better mean value is obtained by $(\mu + \lambda)$-ES approach. Results of PSO (He et al. 2004) and results of UPSOm (Parsopoulos and Vrahatis 2005) are not available for this problem. Evolution of mean error produced by the ABC algorithm for the first 500 cycles is presented on Fig. 2d.

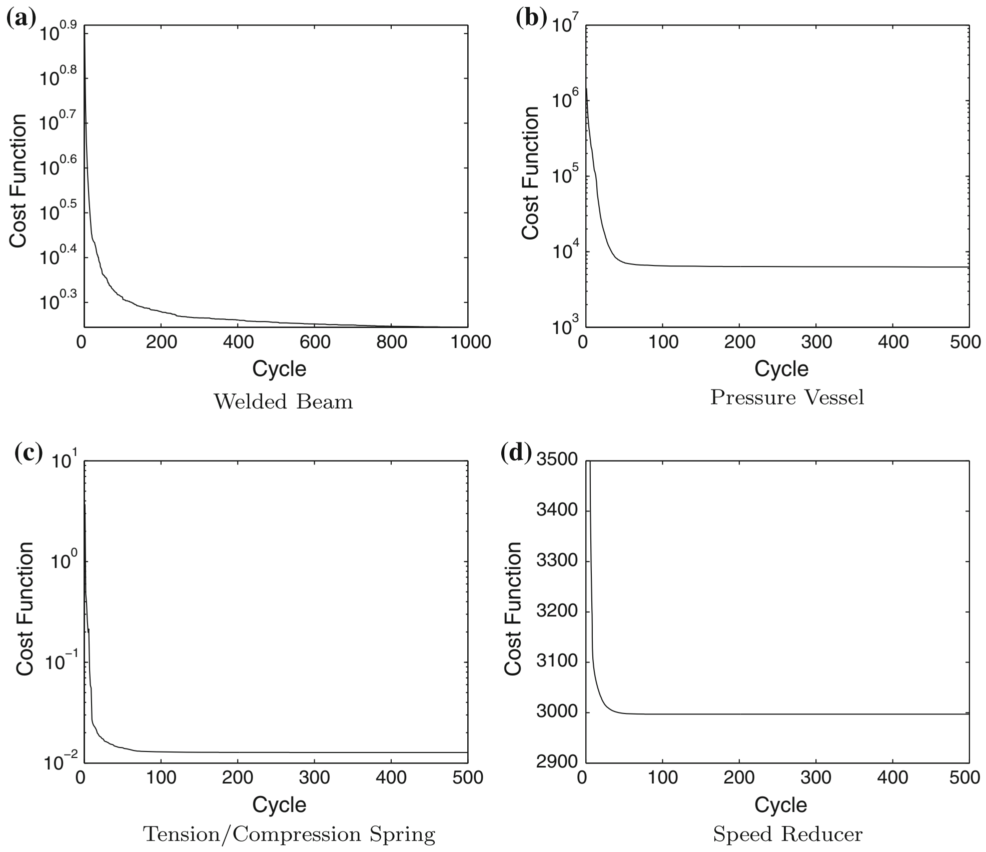

Fig. 2 Evolution of best mean results obtained by ABC algorithm through 30 runs

Table 6 Parameter and constraint values of the best solutions obtained for welded beam problem

	$(\mu + \lambda)$ ES	ABC
$x1$	0.205730	0.205730
$x2$	3.470489	3.470489
$x3$	9.036624	9.036624
$x4$	0.205729	0.205730
$g1$	0.000000	0.000000
$g2$	0.000002	−0.000002
$g3$	0.000000	0.000000
$g4$	−3.432984	−3.432984
$g5$	−0.080730	−0.080730
$g6$	−0.235540	−0.235540
$g7$	−0.000001	0.000000
f(x)	1.724852	1.724852

For gear train problem, results of other algorithms except the UPSOm and ABC algorithms are not available. Results of the UPSOm and ABC algorithms are similar to each other. But the maximum generation number of the UPSOm algorithm is much more than the ABC algorithm. Parameter and constraint values of the best solutions to the problems given in Table 5 are given in Tables 6, 7, 8, 9 and 10.

For handling constraints, algorithms considered in this study use some different techniques. The improved PSO of He et al. (2004) requires a feasible initial population (which, for some problems could be very difficult to get), and it is designed to move only inside the feasible region of a given problem. Therefore, after a solution is produced, it is then checked whether it is feasible or not. If it is not feasible, the previous solution is kept in the population. The ABC algorithm does not need to be initialized by feasible solutions, which is a computationally expensive process. SCA (Ray and Liew 2003) algorithm uses a multilevel Pareto ranking scheme which computes the non-dominated rank of every solution based on the constraint matrix objective vector. For each cluster, a leader is determined by this ranking and is migrated to another society in order to exchange intersociety information. UPSOm (Parsopoulos and Vrahatis 2005) transforms a constrained problem to an unconstrained problem by penalizing the constraint violations. Despite simplicity and direct applicability of the penalty approach, it requires carefully tuning of the weights of the penalty functions. $(\mu + \lambda)$-ES in (Mezura-Montes and Coello Coello 2005b) uses a sorting based handling method assuming that any feasible solution is better than any infeasible one depending on Deb's three

Table 7 Parameter and constraint values of the best solutions obtained for pressure vessel problem

	SCA	PSO	$(\mu+\lambda)$ ES	ABC
$x1$	0.8125	0.8125	0.8125	0.8125
$x2$	0.4375	0.4375	0.4375	0.4375
$x3$	41.9768	42.098446	42.098446	42.098446
$x4$	182.2845	176.636052	176.636596	176.636596
$g1$	−0.0023	0.000000	0.000000	0.000000
$g2$	−0.0370	−0.035881	0.035880	−0.035881
$g3$	−23420.5966	0.000000	0.000000	−0.000226
$g4$	−57.7155	−63.363948	−63.363404	−63.363404
f(x)	6171.0	6059.701610	6059.7143	6059.714339

Table 8 Parameter and constraint values of the best solutions obtained for tension/compression spring problem

	SCA	PSO	$(\mu+\lambda)$ ES	ABC
$x1$	0.0521602	0.051690	0.052836	0.051749
$x2$	0.368159	0.356750	0.384942	0.358179
$x3$	10.648442	11.287126	9.807729	11.203763
$g1$	0.000000	0.000000	−0.000001	−0.000000
$g2$	0.000000	0.000000	0.000000	−0.000000
$g3$	−4.075805	−4.053827	−4.106146	−4.056663
$g4$	−0.719787	−0.727706	−0.708148	−0.726713
f(x)	0.012669	0.012665	0.012689	0.012665

Table 9 Parameter and constraint values of the best solutions obtained for speed reducer problem

	SCA	$(\mu+\lambda)$ ES	ABC
$x1$	3.500000	3.499999	3.499999
$x2$	0.700000	0.699999	0.7
$x3$	17	17	17
$x4$	7.327602	7.300000	7.3
$x5$	7.715321	7.800000	7.8
$x6$	3.350267	3.350215	3.350215
$x7$	5.286655	5.286683	5.287800
$g1$	−0.073915	−0.073915	−0.073915
$g2$	−0.197999	−0.197998	−0.197999
$g3$	−0.493501	−0.499172	−0.499172
$g4$	−0.904644	−0.901472	−0.901555
$g5$	0.000000	0.000000	0.000000
$g6$	0.000633	0.000000	0.000000
$g7$	−0.7025	−0.702500	−0.7025
$g8$	0.000000	0.000000	0.000000
$g9$	−0.583333	−0.583333	−0.583333
$g10$	−0.054889	−0.051325	−0.051326
$g11$	0.000000	−0.010852	−0.010695
f(x)	2994.744241	2996.348094	2997.058412

Table 10 Parameter and constraint values of the best solutions of ABC algorithm obtained for gear train problem

$x1$	$x2$	$x3$	$x4$	f(x)
49	16	19	43	0

feasibility rules. The ABC algorithm also uses these rules in the selection process. However, both algorithms use these rules in different manners; $\mu+\lambda$)-ES algorithm produces λ solutions and sorts $(\mu+\lambda)$ solutions using the three criteria based on feasibility. It makes this selection process at the population level. The ABC algorithm uses these rules while making a selection between the current solution and its neighbour. Moreover, in onlooker phase, the ABC algorithm allows searching the neighbourhood of infeasible solutions by assigning them probability values proportional to their constraint violation values. In (Mezura-Montes and Coello Coello 2005b), the authors argue that $(\mu+\lambda)$-ES approach requires more infeasible solutions in the population. In the ABC algorithm, onlooker and scout processes may provide the algorithm to have infeasible solutions. Hence, this improves the exploration capability of the algorithm and maintains diversity in the population.

In terms of the complexity of the algorithms, while UPSO, ABC and $(\mu+\lambda)$-ES have similar complexities, PSO of He et al. (2004) has more complexity since it requires a feasible initial population and SCA adds extra computational cost derived of clustering routines (Mezura-Montes and Coello Coello 2005b).

Conclusion

In this work, the performance of Artificial Bee Colony algorithm was compared to that of the Differential Evolution and Particle Swarm Optimization algorithms on unconstrained large-scale well-known benchmark problems. It can be concluded that the ABC algorithm has superior performance to the DE and PSO algorithms on large-scale unconstrained optimization problems since the ABC algorithm balances exploration and exploitation processes and employs different selection operators together: greedy selection, probabilistic selection and random selection.

Engineering design problems are generally nonlinear and constrained optimization problems. Therefore, in order to solve these problems a global optimization algorithm is required. In this work, a modified version of the ABC algorithm has been proposed to solve engineering design problems. From the studies, it can be concluded that ABC algorithm is a promising tool for optimizing constrained engineering problems.

Acknowledgement This work was supported by Erciyes University, the Scientific Research Projects Unit under contract FBA–06–22.

Appendix: Constrained engineering design problems

Problem 1: the welded beam problem

Welded beam design illustrated in Fig. 3 minimizes the cost of the beam subject to constraints on shear stress, τ , bending stress in the beam, σ , buckling load on the bar, P_c, end deflection of the beam, δ, and side constraints. There are four design parameters x_1, x_2, x_3 and x_4 correspond to h, l, t and b variables, respectively, shown in Fig. 3.

The detailed formulation of the problem is as follows:

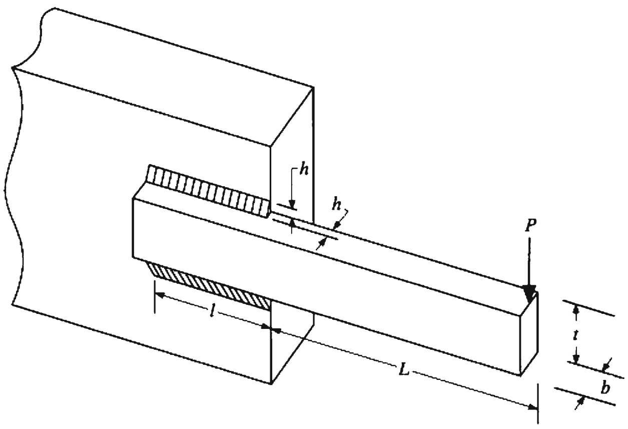

Fig. 3 The welded beam problem

$$
\begin{aligned}
\min_{\mathrm{x}}\ & f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \\
\text{subject to}\ & g_1(X) : \tau(x) - \tau_{\max} \leq 0 \\
& g_2(X) : \sigma(x) - \sigma_{\max} \leq 0 \\
& g_3(X) : x_1 - x_4 \leq 0 \\
& g_4(X) : 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) \\
& \qquad -5.0 \leq 0 \\
& g_5(X) : 0.125 - x_1 \leq 0 \\
& g_6(X) : \delta(x) - \delta_{\max} \leq 0 \\
& g_7(X) : P - P_c(x) \leq 0
\end{aligned}
$$

where $\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x2}{2R} + (\tau'')^2}$, $\tau' = \frac{P}{\sqrt{2}x_1x_2}$, $\tau'' = \frac{MR}{J}$,

$$
M = P\left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},
$$

$$
J = 2\left\{\frac{x_1x_2}{\sqrt{2}}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},
$$

$$
\sigma(X) = \frac{6PL}{x_4x_3^2}, \delta(X) = \frac{4PL^3}{Ex_3^3x_4},
$$

$$
P_c = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)
$$

where $P = 6000$ lb., $L = 14$ in, $\delta_{\max} = 0.25$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, $\tau_{\max} = 1,3600\,psi$, $\sigma_{\max} = 3,0000$ psi, $X = (x_1, x_2, x_3, x_4)^{\mathrm{T}}$, $0.1 \leq x_1, x_4 \leq 2.0$, $0.1 \leq x_2, x_3 \leq 10$.

Problem 2: the pressure vessel problem

Second example is minimization of the total cost comprising of material, forming and welding costs of a cylindrical vessel as shown in Fig. 4. The four design variables are x_1 (thickness T_S of the shell), x_2 (thickness T_H of the head), x_3 (inner radius R) and x_4 (length L of the cylindrical section of the vessel, not including the head). x_1 and x_2 are to be in integral multiples of 0.0625 inch which are the available thicknesses of rolled steel plates. The radius x_3 and the length x_4 are continuous variables.

$$
\begin{aligned}
\min_{X}\ & f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 \\
& \qquad +19.84x_1^2x_3 \\
\text{subject to}\ & g_1(X) : -x_1 + 0.0193x_3 \leq 0 \\
& g_2(X) : -x_2 + 0.00954 \leq 0 \\
& g_3(X) : -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
& g_4(X) : x_4 - 240 \leq 0
\end{aligned}
$$

where $X = (x_1, x_2, x_3, x_4)^{\mathrm{T}}$. The ranges of the design parameters are $0 \leq x_1, x_2 \leq 99$, $10 \leq x_3, x_4 \leq 200$.

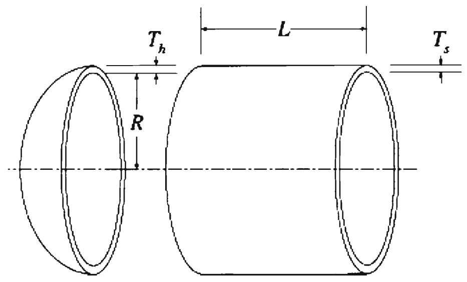

Fig. 4 The pressure vessel problem

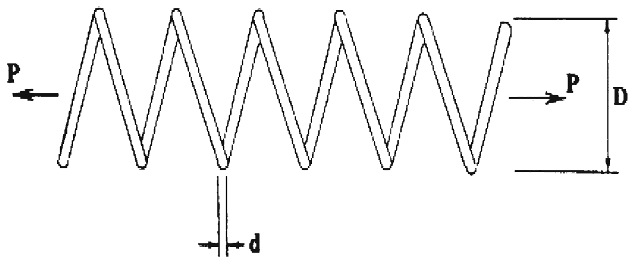

Fig. 5 The tension/compression spring problem

Problem 3: the tension/compression spring problem

The tension/compression problem deals with the minimization of the weight of the tension/compression spring shown in Fig. 5, subject to constraints on the minimum deflection, shear stress, surge frequency, diameter and design variables.

The design variables are the wire diameter, $d(x_1)$, the mean coil diameter, $D(x_2)$, and the number of active coils, $N(x_3)$. The problem is formulated as:

$$\min_X f(X) = (N+2)Dd^2$$

$$\text{subject to } g_1(X): 1 - \frac{D^3N}{71785d^4} \le 0$$

$$g_2(X): \frac{4D^2-dD}{12566(Dd^3-d^4)} + \frac{1}{5108d^2} - 1 \le 0$$

$$g_3(X): 1 - \frac{140.45d}{D^2N} \le 0$$

$$g_4(X): \frac{D+d}{1.5} - 1 \le 0$$

$X = (d, D, N)^{\mathrm{T}}$, $0.05 \le d \le 2.0$, $0.25 \le D \le 1.3$, $2.0 \le N \le 15.0$

Problem 4: speed reducer design

The aim of the speed reducer design shown in Fig. 6 is to minimize the weights of the speed reducer subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts. Design parameters of the speed reducer problem, the face width (b), module of teeth (m), number of teeth in the pinion (z), length of the first shaft between bearings (l_1), length of the second shaft between bearings (l_2) and the diameter of the first shaft (d_1) and second shaft (d_2) correspond to $x_1, x_2, \ldots, x_7$, respectively. Some optimization algorithms have been reported to have difficulties in finding the feasible space and is an example of a mixed integer programming problem. The third variable (number of teeth) is of integer value while all other variables are continuous (Mezura-Montes and Coello Coello 2005b).

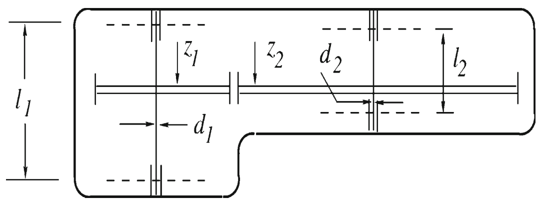

Fig. 6 The speed reducer problem

$$f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3)$$

$$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \le 0$$

$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0$$

$$g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0$$

$$g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0$$

$$g_5(x) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6\right)^{1/2}}{110.0x_6^3} - 1 \le 0$$

$$g_6(x) = \frac{\left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 157.5\times10^6\right)^{1/2}}{85.0x_7^3} - 1 \le 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \le 0$$

$$g_8(x) = \frac{5x_2}{x_1} - 1 \le 0$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \le 0$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

where $2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.8 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9, 5.0 \le x_7 \le 5.5$.

Problem 5: gear train design

Gear train design aims to minimize the cost of the gear ratio of the gear train as shown in Fig. 7.

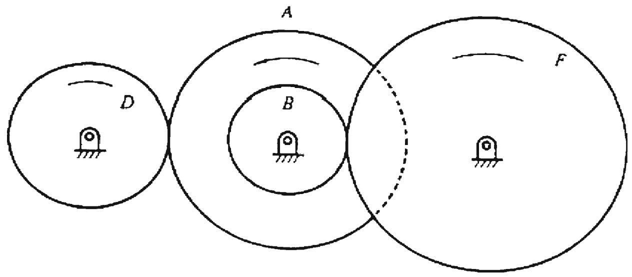

Fig. 7 The gear train problem

The gear ratio is defined as:

$$\text{gear ratio} = \frac{n_B n_D}{n_F n_A}$$

The design variables of the problem n_A, n_B, n_D, n_F , will be denoted as x_1, x_2, x_3, x_4 , respectively, are all integers in the range [12,60]. The problem is formulated as:

$$\min_X f(X) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2$$
$$\text{subject to } 12 \leq x_i \leq 60, \quad i = 1, \ldots, 4$$

References

Bartak, R., Salido, M. A., & Rossi, F. (2008). Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing*, doi:10.1007/s10845-008-0203-4.

Basturk, B. & Karaboga, D. (2006). An artificial bee colony (abc) algorithm for numeric function optimization. In *IEEE Swarm Intelligence Symposium 2006 Indianapolis*. Indiana, USA.

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

Boyer, D. O., Martnez, C. H., & Pedrajas, N. G. (2005). Crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research, 24*, 1–48.

Coello Coello, C.A. (2002). Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering, 191*(1112), 1245–1287.

Coello Coello, C. A. (1999). *A survey of constraint handling techniques used with evolutionary algorithms*. Technical report, LANIA, LaniaRI9904.

Corne, D., Dorigo, M., & Glover, F. (1999). *New ideas in optimization*. New York: McGraw-Hill.

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

Digalakis, J. G., & Margaritis, K. G. (2002). An experimental study of benchmarking functions for genetic algorithms. *International Journal of Computer and Mathematics, 79*(4), 403–416.

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

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

Joines, J., & Houck, C. (1994). 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*, (pp. 579–584). Orlando, Florida: IEEE Press.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*. Technical report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D., & Akay, B. (2007). An artificial bee colony (abc) algorithm on training artificial neural networks. In *15th IEEE Signal Processing and Communications Applications, SIU 2007* (pp.1–4). Eskisehir, Turkiye.

Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation, 214*, 108–132.

Karaboga, D., Akay, B., & Ozturk, C. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In *Modeling decisions for artificial intelligence* (vol. 4617/2007 of LNCS, pp. 318–329). Springer.

Karaboga, D. & Basturk, B. (2007a). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In *Advances in soft computing: foundations of fuzzy logic and soft computing* (vol. 4529/2007 of LNCS, pp. 789–798). Springer.

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

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

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks* (vol. 4, pp. 1942–1948).

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

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

Mezura-Montes, E., & Coello Coello, C. A. (2005b). Useful infeasible solutions in engineering optimization with evolutionary algorithms. In *MICAI 2005: Advances in Artificial Intelligence of Lecture Notes in Computer Science* (vol. 3789/2005, pp. 652–662). Berlin: Springer.

Michalewicz, Z., & Attia, N. F. (1994). Evolutionary optimization of constrained problems. In *Proceedings of the 3rd Annual Conference on Evolutionary Programming* (pp. 98–108). World Scientific, Singapore

Michalewicz, Z., Deb, K., Schmidt, M., & Stidsen, T. (1999). Evolutionary algorithms for engineering applications. In K. Miettinen, P. Neittaanmäki, M. M. Mäkelä, & J. Périaux (Eds.), *Evolutionary algorithms in engineering and computer science* (pp. 73–94). Chichester: Wiley.

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

Michalewicz, Z. & Nazhiyath, G. (1995). 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* (pp. 647–651). Piscataway, New Jersey: IEEE Press.

Myung, H., Kim, J. -H., & Fogel, D. B. (1995). 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* (pp. 449–463). Cambridge, Massachusetts: MIT Press.

Paredis, J. (1994). Co-evolutionary constraint satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature* (pp. 46–55). New York: Springer.

Parmee, I. C., & Purchase, G. (1994). 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* (pp. 97–102). Plymouth, UK: University of Plymouth.

Parsopoulos, K., & Vrahatis, M. (2002). Particle swarm optimization method for constrained optimization problems. In: P. Sincak, J. Vascak, V. Kvasnicka, & J. Pospichal (Eds.) *Intelligent technologies—theory and application: New trends in intelligent technologies, volume 76 of frontiers in artificial intelligence and applications*, (pp. 214–220). Amsterdam: IOS Press.

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

Powell, D., & Skolnick, M. M. (1993). 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)* (pp. 424–431). San Mateo, California. University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers.

Rao, S. S. (1996). *Engineering optimization*. New York: Wiley.

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

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

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

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

Smith, A. E., & Coit, D. W. (1997). *Constraint handling techniques-penalty functions in handbook of evolutionary computation (vol. 5.2)*. New York: Oxford University Press and Institute of Physics.

Storn, R., & Price, K. (1995). *Differential evolution -a simple and efficient adaptive scheme for global optimization over continuous spaces*. Technical report, International Computer Science Institute, Berkley.

Yeniay, O. (2005). Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications, 10*, 45–56.

Zhang, J. Y., Liang, S. Y., Yao, J., Chen, J. M., & Huang, J. L. (2006). Evolutionary optimization of machining processes. *Journal of Intelligent Manufacturing, 17*, 203–215.