An International Journal of Optimization and Control: Theories & Applications
ISSN:2146-0957 eISSN:2146-5703
Vol.7, No.1, pp.98-111 (2017)
http://doi.org/10.11121/ijocta.01.2017.00342

RESEARCH ARTICLE

Artificial bee colony algorithm variants on constrained optimization

Bahriye Akay*, Dervis Karaboga

Department of Computer Engineering, Erciyes University, Kayseri, Turkey
bahriye@erciyes.edu.tr, karaboga@erciyes.edu.tr

ARTICLE INFO

Article History:
Received 29 September 2016
Accepted 15 November 2016
Available 12 January 2017

Keywords:
Artificial bee colony algorithm
Constrained optimization
Deb's rules

AMS Classification 2010:
65K10

ABSTRACT

Optimization problems are generally classified into two main groups: unconstrained and constrained. In the case of constrained optimization, special techniques are required to handle with constraints and to produce solutions in the feasible space. Intelligent optimization techniques that do not make assumptions on the problem characteristics are preferred to produce acceptable solutions to the constrained optimization problems. In this study, the performance analysis of artificial bee colony algorithm (ABC), one of the intelligent optimization techniques, is examined on constrained problems and the effect of some modifications on the performance of the algorithm is examined. Different variants of the algorithm were proposed and compared in terms of efficiency and stability. Depending on the results, when DE operators were integrated into ABC algorithm, an enhancement in the performance was gained in addition to preserving the stability of the basic ABC. The ABC algorithm is a simple optimization algorithm that can be efficiently used for constrained optimization without requiring a priori knowledge.

1. Introduction

Design problems are optimization problems in which the parameters of the system are decided in order to obtain the best performance. Formulating design problems as optimization problems and solving them by using an appropriate optimization tool minimizes the experimental costs and errors. In most of the design problems, problems have some constraints that the solutions must satisfy.

Traditional optimization algorithms are not successful when the problems are nonlinear and have many constraints and discrete variables. Evolutionary algorithms (EAs) [1] that mimic the genetic inheritance and natural selection do not make assumptions on the problem characteristics and can be used for constrained, nonlinear design problems successfully [2]. In EAs, a population of solutions are assigned fitness values and new child solutions are produced after reproduction, mutation, crossover operators. Better solutions are retained in the population applying a selection operator and the population quality is increased which means convergence to better solutions.

Swarm intelligence (SI) based algorithms employ similar operators to those of evolutionary algorithms while SI algorithms are also capable of using collective intelligence coming due to the interactions of individuals in the swarm. Genetic algorithm [3], Differential Evolution [4] are typical examples of EAs; Ant Colony Optimization [5], Particle Swarm Optimization [6] and Artificial Bee Colony [7] algorithms are examples of SI based algorithms.

All these algorithms were initially proposed for unconstrained optimization and their basic versions did not have any mechanism to produce solutions in feasible space or to prefer feasible solution to infeasible one. Some operators are integrated to the algorithms in order to search the feasible space and solve constrained problems such as

*Corresponding Author

transforming operator that preserve feasibility of solutions, penalty functions to penalize constraint violations in the cost function, distinction operator that separate feasible and infeasible, hybrid methods that incorporates evolutionary methods and deterministic procedures [8]. Artificial Bee Colony Algorithm proposed for constrained optimization [9, 10] prefers feasible solutions to infeasible solutions applying Deb's rules [11] in the selection phase. Another modification in basic ABC concerns fitness assignment scheme which considers both feasible solutions and infeasible solutions because the population can include both types of solutions to maintain diversity in the population.

Although there are some studies related to ABC on the constrained optimization, effect of the neighbor production mechanism has not been studied to analyze how they cover feasible space or whether they are capable of making distinction between feasible and infeasible space or which one is more efficient for constrained optimization. In this study, various ABC algorithm variants, which employ different neighbor productions mechanisms and neighborhood topologies, are analyzed in terms of success rates. Effect of the neighbor solution production in the search behavior of the algorithms and in the convergence are investigated. Basic ABC algorithm for unconstrained optimization and proposed ABC algorithm variants for constrained optimization are summarized in the second and the third sections, respectively. In the forth section, the experimental studies are presented and the results are discussed. The last section is dedicated to the conclusion.

2. Artificial bee colony algorithm

Honey bees perform many intelligent tasks and foraging is one of the most important tasks in the colony. The foraging is performed using the collective intelligence based on the communication and interaction of the bees. Bees are specialized depending on the task they are assigned to. In a real colony, there are three types of bees assigned to the foraging task: employed bees, onlooker bees and scout bees. The scout bees search for new food sources of which the nectar will be loaded to the hive. Discovered sources are exploited by the employed bees and the employed bees share information with the onlooker bees who wait in the hive. Onlooker bees select a food source to fly based on the information taken from the employed bees. When a source is exhausted, it is abandoned and its bee switches her role to a scout bee. In the foraging task, bees try to maximize the nectar amount loaded to hive by finding the most profitable sources in the environment.

ABC algorithm [7] simulates the foraging behavior of honey bees in nature. Locations of the food sources correspond to the parameters of the problem and finding the location of the most profitable source is an optimization problem. Nectar amount of the solution is measured by the fitness of a parameter vector. Main steps of the algorithm are given in Alg. 1:

Algorithm 1. *Pseudo-code of ABC algorithm*

```
1: Initialization
2: Evaluation
3: cycle=1
4: repeat
5:    Employed Bees Phase
6:    Calculate Probabilities for Onlookers
7:    Onlooker Bees Phase
8:    Scout Bees Phase
9:    Memorize the best solution so far
10:   cycle=cycle+1
11: until cycle=Maximum Cycle Number
```

In the initialization phase of the algorithm, values are passed to the control parameters: the number of food sources (CS), maximum cycle number (MCN) and the limit that controls whether a source is exhausted or not. If the problem has D parameters to be optimized, a population of CS number of D dimensional solutions are generated randomly using Eq. 1.

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{\min}^j) \qquad (1)$$

where x_{min}^j and x_{max}^j are lower and upper bound of the parameter j, respectively.

In the evaluation phase, each solution is substituted in the cost function and assigned a fitness value using the cost function associated with the problem. Subsequently, the employed bee, onlooker bee and scout bee phases are repeated until the cycle number reaches to MCN.

In the employed bee phase, a local search is carried out in the vicinity of each solution by Eq. 2:

$$x'_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \qquad (2)$$

where $k \neq i$ is a neighbor solution drawn randomly from the population, ϕ_{ij} is random real number within the range [-1,1] drawn from uniform distribution. After new solution is assigned a fitness value, ABC algorithm performs greedy selection by which if the fitness of $\vec{x}'$ is better than the fitness of $\vec{x}$, $\vec{x}$ is discarded and $\vec{x}'$ is included in the population. Otherwise, $\vec{x}$ is retained in the population. To be used in the scout bee phase, a counter associated with each solution holds the number of times that the solution is retained in the population. In the onlooker bee phase, onlooker bees use the information obtained from employed bees and select high quality sources to fly. Each solution is assigned a probability value (Eq. 3) proportional to its fitness value obtained from the employed bee phase. In basic ABC algorithm, a roulette wheel selection scheme is applied to select potentially high quality solutions but also to give chance to low quality solutions to be selected.

$$p_i = \frac{fitness_i}{\sum\limits_{i=1}^{CS} fitness_i} \tag{3}$$

Once a solution is determined using roulette wheel selection, a local search is performed using Eq. 2. As in the employed bee phase, a greedy selection is applied and counters are updated. The onlooker bee phase promotes more local searches around the better solutions which is a positive feedback feature.

In the scout bee phase, counters associated with each solution are checked and if a counter is higher than the control parameter limit, that source is assumed to be exhausted. The exhausted source is replaced with a new solution produced by Eq. 1 and its counter is set to 0. This phenomena arises a negative feedback effect and fluctuation in the algorithm which avoids to trap local minima.

The algorithm performs a balanced exploration and exploitation and has the advantage of employing less control parameters. ABC algorithm has been applied to many problems in wide range of fields [12, 13, 14] and has shown superior performance on high dimensional multimodal problems [15, 9]

3. Artificial bee colony algorithm for constrained optimization

In most of the design problems, problems have some constraints that the solutions must satisfy. A constrained design problem is defined as in Eq. 4.

$$\begin{array}{ll} \text{minimize } f(\vec{x}), & \vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n \\ & l_i \le x_i \le u_i, \\ & i = 1, \ldots, n \\ \text{subject to :} & g_j(\vec{x}) \le 0, \\ & \text{for } j = 1, \ldots, q \\ & h_j(\vec{x}) = 0, \\ & \text{for } j = q+1, \ldots, m \end{array} \tag{4}$$

where f is defined in n-dimensional search space, (S), and each $\vec{x_i}$ parameter is bounded by the range $[l_i, u_i]$. Constrained optimization finds a parameter vector $\vec{x}$ which minimizes the cost function,$(f(\vec{x}))$ and does not violate inequality $(g_j(\vec{x}))$ and equality $(h_j(\vec{x}))$ constraints. A feasible solution satisfies all constraints and in feasible space $F \subseteq S$, which is defined by $m \ge 0$ constraints [16]. Like the other EAs, ABC algorithm was initially proposed for unconstrained optimization and some modifications have to be made in the algorithm in order to cope with the constraints and provide solutions in the feasible space (F). In the subsequent sections, proposed ABC algorithm variants for constraint optimization are explained.

3.1. ABCV1: ABC algorithm for constrained optimization

ABC algorithm for constrained optimization [9, 10] uses the same framework with the basic ABC algorithm. ABCV1 includes employed bees, onlooker bees and scout bees phases as well while there have been some modifications inside the phases. ABCV1 does not need initial solutions to be in feasible space and initial solutions are produced using Eq. 1 as in basic ABC algorithm.

In the employed bee phase of ABCV1, a local search is conducted in the neighborhood of the solution in bee's memory using Eq. 5 instead of Eq. 2. Eq. 2 makes modification on one dimension of the current solution while Eq. 5 makes modification on dimensions if a uniformly distributed random number R_j is lower than perturbation rate, MR. MR is a control parameter introduced in ABCV1. High values of MR speeds up the convergence but has negative effect on fine tuning.

$$v_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) & , \ if \ \ R_j < MR \\ x_{ij} & , \ \text{otherwise} \end{cases} \tag{5}$$

After generating a mutant solution v_{ij} for each solution, instead of greedy selection, ABCV1 applies Deb's rules [11] that are listed below:

- A feasible solution ($violation_i \leq 0$) is chosen against an infeasible solution ($violation_j > 0$) (solution i is dominant),
- If both of the solutions are feasible($violation_i \leq 0$, $violation_j \leq 0$), the one with better objective function value is chosen ($f_i < f_j$, solution i is dominant),
- If both of the solutions are infeasible ($violation_i > 0$, $violation_j > 0$), the one with smaller constraint violation is chosen ($violation_i < violation_j$, solution i is dominant).

In the constrained optimization, the cost function value obtained evaluating an infeasible solution might be lower than the one obtained with a feasible solution. In the probability assignment scheme, the feasible solutions should have higher fitness values than infeasible solutions to promote feasible regions. Therefore, the probability of the feasible solutions starts from 0.5 (within the range [0.5,1])and the probability of the infeasible solutions are assigned within the range [0,0.5] based the violation of the solutions. Eq. 6 can be used for this purpose:

$$p_i = \begin{cases} 0.5 + \left(\frac{fitness_i}{\sum\limits_{j=1}^{CS} fitness_j} \right) * 0.5 & if\ feasible \\ \left(1 - \frac{violation_i}{\sum\limits_{j=1}^{CS} violation_j} \right) * 0.5 & if\ infeasible \end{cases} \tag{6}$$

Eq. 6 assigns higher values to the feasible solutions but the solutions with lower probability value have also chance to be chosen by the roulette wheel selection. This property provides population diversity and search ability near the boundary lines. In order to avoid feasible solutions to be discarded quickly in the scout phase, the limit checking is performed in each SPP cycles instead of each cycle. SPP is also another control parameter introduced in ABCV1.

3.2. ABCV2: Each parameter from different solution

In ABCV2, Eq. 5 is replaced with a different search operator defined by Eq 7. Eq. 5 exploits kth solution for all dimensions while Eq 7 uses a different neighbor (k_j) for each dimension so that information of more individuals are spread and more interaction can be obtained from the population. For each parameter, a random number is drawn and a random neighbor is selected. If the random number is less than MR, Eq. 7 changes this parameter using the information of the neighbor selected.

$$x'_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{k_j j} - x_{ij}), & R_j < MR \\ x_{ij} & otherwise \end{cases} \tag{7}$$

The other parts of the algorithm are retained as in ABCV1.

3.3. ABCV3: ABC algorithm utilizing individuals in a neighborhood topology

In ABCV1 and ABCV2, selected neighbors are drawn from the population in a global manner without considering any criterion between the solutions. In ABCV3, it is proposed to select the neighbors from a neighborhood topology which comprises the solutions within a predefined radius. Neighborhood of i the solution, N_i, is defined by the expression given by Eq. 8:

$$N_i = \left\{ \bigcup_{k \neq i} x_k \, | d_{ik} \leq d_{avg} \right\} \tag{8}$$

As seen from the expression, a solution x_k, is assumed to be a neighbor of the current solution, x_i, if the distance between them (d_{ik}) is less than the average Euclidian distance (d_{avg}). In ABCV3, Eq. 9 is used to produce a new solution:

$$x'_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{k_j j} - x_{ij}), & R_j < MR, \quad \text{and } \mathrm{x_{k_j} \in N_i} \\ x_{ij}, & otherwise \end{cases} \tag{9}$$

3.4. ABCV4:ABC algorithm using a neighborhood topology in the Onlooker bee phase

The quick ABC algorithm [13] is proposed to enhance the local search capability of the basic ABC algorithm. In the basic ABC algorithm, an onlooker bee performs perturbation the solution selected based on the roulette wheel selection while in quick ABC algorithm, an onlooker bee performs on the best solution of a neighborhood defined by a radius, NR.

3.5. ABCV5,ABCV6,ABCV7:ABC algorithms using the mutation and crossover operators of differential evolution algorithm

Differential Evolution algorithm [4], proposed by Storn and Price, is an iterative and population-based optimization algorithm for optimizing the continuous functions. It is a fast, simple and easy applicable algorithm. As in the other evolutionary algorithms, it has evaluation, mutation, crossover and selection phases. In mutation phase, for each solution, a mutant solution is produced by weighing the difference of the solutions chosen randomly (Eq. 10):

$$\hat{x}_i = x_{r_1} + F(x_i - x_{r_2}) \tag{10}$$

where $r1 \neq r2 \neq i$ and F is real valued scaling factor within the range [0,2]. The mutant solution is subjected to the crossover operation with the original solution (Eq. 11):

$$y_i = \begin{cases} \hat{x}_i, & R_j \leq C_R \\ x_i, & R_j > C_R \end{cases} \tag{11}$$

where C_R is crossover rate, R_j is a random real value drawn within the range [0,1]. New solution is evaluated using the cost function and greedy selection operator is applied to decide whether the original solution or new solution will be retained in the population. These steps are repeated until the termination criteria is satisfied. Although DE algorithm produces efficient results on some functions, the statistics related to its stability such as mean value and standard deviation are not satisfactory [9]. For this reason, DE's operators are integrated with ABC algorithm in order to combine DE's success on unimodal functions and to preserve stability of the ABC algorithm. To achieve this, Eq. 5 in ABC is replaced with Equations 10 and 11. This modification is carried out only in employed bee phase in ABCV5, only is onlooker bee phase in ABCV6 and both in employed bee phase and in onlooker bee phase in ABCV7 variant.

3.6. ABCV8:ABC algorithm using the global best

In order to include the information due to the best solution into the neighborhood, Eq. 12 is introduced which also uses the weighted difference of the current solution and global best solution, *gbest*.

$$x'_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{kj} - x_{ij}) + \theta_{ij}(gbest_j - x_{ij}), & R_j < MR \\ x_{ij} & otherwise \end{cases} \tag{12}$$

where θ_{ij} is a random real number within the range [-1,1] drawn from uniform distribution.

3.7. ABCV9:ABC algorithm with adaptive Scout behavior

In the basic ABC algorithm, the limit value to abandon a food source is fixed for all solutions during the population evolution. In this proposed version, each food source is assigned a counter value which is proportional to its fitness value if the solution is feasible and disproportional to its constraint violation amount if the solution is infeasible. This gives more search opportunity in the locality of the high quality solutions. For this purpose, the counter of each solution is divided by the probability value defined by Eq. 6 (Eq. 13)

$$counter_i = \frac{counter_i}{p_i} \tag{13}$$

Vicinity of the better solutions have higher chance to be explored compared to the others.

4. Experimental study and results

In this study, different variants of the ABC algorithm are analyzed on constrained optimization test problems. The details of the thirteen test problems used are given at the end of the paper and characteristics of the problems are given in Table 1.

The common control parameters of the ABC algorithm variants are the number food sources (CS) that are exploited during search, the maximum number of cycles (MCN) that the phases are repeated for, modification rate (MR) that control the number of parameters to be perturbed, *limit* that is the maximum number of exploitations a solution allowed and scout production period (SPP) that the limit checking period and neighborhood radius (NR) for the variants that use a neighborhood topology. The values set for these parameters are given in Table 2.

All implementations were developed using Delphi 7 programming language and run on a computer with 64 bit 3.06 GHz Intel processor and 8 GB of RAM. All algorithms were run 30 times with different seed numbers and statistics of the results (the best, mean, worst and standard deviation) of 30 runs are summarized in Tables 3-11 for the versions ABCV1-ABCV9, respectively. The results of the basic ABC algorithm proposed for constrained optimization (ABCV1) are taken from the study in ref. [10]. ABCV1 is taken as the control algorithm to test the efficiency of the variants. The best result statistic can be used to test the efficiency while the mean and standard

Table 1. Characteristics of the test problems used in the experiments. D:Dimension of the problem, LI: Linear inequality, NI: Nonlinear inequality, LE: Number of linear equalities, NE: Number of nonlinear equalities, *rho*: the ratio of feasible region over the search space [17].

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

Table 2. Values of the control parameters used in the experiments

CS	MCN	MR	$limit$	SPP	NR
40	6000	0.8	$CSxD$	$CSxD$	1

Table 3. The results of ABCV1 [10].

Problem	Optimum	Best	Mean	Worst	StdDev
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	0.75	0.75	0.75	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.760	0.968	1.000	0.055

deviation can be used as an indicator for the stability and robustness of the algorithm.

The results of ABCV2 in which each parameter of a mutant solution is taken from a different neighbor are given in Table 4. When the best results are investigated, an important improvement is achieved on g13 function while the performance on g10 gets worsened. When the mean and standard deviation results are analyzed, it is seen that the stability of ABCV2 is worse than ABCV1.

ABCV2 uses a global neighborhood while ABCV3 chooses the neighbors in a local topology. It is seen that the results in Table 5 are similar to the results in Table 3. Because the genotypes of the solutions in a local neighborhood resemble each other in later cycles, choosing the parameters from different neighbors does not have significant effect compared to choosing all the parameters from the same solution.

The results of ABCV4 in Table 6 suggest that there is no significant improvement when the selection strategy is changed in the onlooker bee phase of ABC algorithm. That means when an onlooker bee performs perturbation on the best solution in a neighborhood instead of a solution chosen using roulette wheel selection does not affect the algorithm performance on constrained optimization. However, it can be said that it has positive effect in the convergence rate of the algorithm in the earlier cycles.

Table 4. The results of ABCV2 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-15.000	-15.000	0.000
g02	0.803619	0.803602	0.801538	0.792962	0.003351
g03	1.000	1.000	0.988920	0.941852	0.016025
g04	-30665.539	-30665.539	-30665.539	-30665.539	0.000
g05	5126.498	5125.225	5170.748	5363.876	56.745
g06	-6961.814	-6961.814	-6961.814	-6961.814	0.000
g07	24.306	24.404	24.578	24.971	0.149894
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.634	680.644	680.667	0.009027
g10	7049.25	7110.214	7325.496	7655.235	125.834
g11	0.75	0.75	0.750112	0.751575	0.000356
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.071093	0.326423	0.476347	0.095829

Table 5. The results of ABCV3 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-15.000	-15.000	0.000
g02	0.803619	0.803600	0.793502	0.774616	0.007767
g03	1.000	1.000	0.998776	0.934880	0.013489
g04	-30665.539	-30665.539	-30665.539	-30665.539	0.000
g05	5126.498	5126.491	5206.871	5522.573	98.799
g06	-6961.814	-6961.813	-6961.809	-6961.792	0.005644
g07	24.306	24.351	24.476	24.781	0.106532
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.633	680.640	680.649	0.003598
g10	7049.25	7061.397	7235.247	7440.239	117.820
g11	0.75	0.75	0.750001	0.750016	0.000002
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.774675	0.975163	1.030087	0.050736

Table 6. The results of ABCV4 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-15.000	-15.000	0.000
g02	0.803619	0.803573	0.789310	0.760181	0.012939
g03	1.000	1.000	0.985268	0.964150	0.009554
g04	-30665.539	-30665.539	-30665.539	-30665.539	0.000
g05	5126.498	5126.495	5198.656	6112.214	176.557
g06	-6961.814	-6961.796	-6961.482	-6960.889	0.223100
g07	24.306	24.328	24.497	24.989	0.122971
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.632	680.646	680.666	0.007796
g10	7049.25	7051.682	7249.228	7494.751	105.537
g11	0.75	0.75	0.750016	0.750193	0.000037
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.654826	0.778555	0.873567	0.135475

In ABC variants, ABCV5, ABCV6 and ABCV7 the search operator of DE algorithm is integrated into employed bee, onlooker bee and both phases and the results are presented in Tables 7-9, respectively. When the best solutions are investigated, it can be seen that on twelve problems g1-g12, these three variants reach the optimum. On

Table 7. The results of ABCV5 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-14.894270	-13.000	0.409793
g02	0.803619	0.803618	0.780956	0.744360	0.016786
g03	1.000	1.000	0.998049	0.968504	0.006637
g04	-30665.539	-30665.539	-30665.539	-30665.539	0.000
g05	5126.498	5126.484	5298.683	5889.126	224.498
g06	-6961.814	-6961.814	-6961.814	-6961.814	0.000
g07	24.306	24.306	24.321	24.435	0.028042
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.630	680.630	680.631	0.000454
g10	7049.25	7049.253	7091.954	7242.825	65.170
g11	0.75	0.75	0.75	0.75	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.079478	0.484924	0.764011	0.134548

Table 8. The results of ABCV6 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-14.866	-13.000	0.498887
g02	0.803619	0.803618	0.780261	0.712447	0.020249
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	5327.777	5809.034	220.034
g06	-6961.814	-6961.814	-6961.814	-6961.814	0.000
g07	24.306	24.306	24.317	24.384	0.019548
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.630	680.630	680.632	0.000655
g10	7049.25	7049.255	7082.506	7250.971	51.227
g11	0.75	0.75	0.75	0.75	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.096854	0.419242	0.911652	0.178598

Table 9. The results of ABCV7 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-14.619	-11.281	1.021
g02	0.803619	0.803591	0.644725	0.471498	0.084904
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	5339.028	5914.850	234.484
g06	-6961.814	-6961.814	-6961.814	-6961.814	0
g07	24.306	24.306	24.372	24.760	0.110926
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.630	680.632	680.642	0.003779
g10	7049.25	7049.251	7098.076	7469.046	107.195
g11	0.75	0.75	0.75	0.75	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.058446	0.378819	0.752359	0.157990

g13 problem, only ABCV7, in which the operator is used in both employed bee and onlooker bee phases, reaches the optimum.

The results of ABCV8 in which the global best solution is exploited are presented in Table 10. In terms of the best results in the table, the results

Table 10. The results of ABCV8 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-15.000	-15.000	0.000
g02	0.803619	0.803610	0.788472	0.681864	0.022395
g03	1.000	1.000	1.000	1.000	0
g04	-30665.539	-30665.539	-30665.539	-30665.539	0.000
g05	5126.498	5126.834	5238.826	5463.804	95.606
g06	-6961.814	-6961.806	-6961.686	-6961.393	0.095273
g07	24.306	24.327	24.462	24.714	0.109640
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.631	680.636	680.641	0.002814
g10	7049.25	7049.997	7139.028	7426.031	71.411
g11	0.75	0.75	0.75	0.75	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.494877	0.975300	1.000	0.091426

are similar to those of ABCV1 except for g13. In addition, stability of ABCV8 is also close to the basic version.

The results of the ABC algorithm with adaptive scout behavior (ABCV9) are presented in Table 11. This modification does not provide any improvement or change in the performance of the basic algorithm.

In addition to comparing the performance of the variants with control algorithm ABCV1, an overall comparison is conducted in terms of the best results in Table 12. If an algorithm is the best among all the algorithms on a problem, it is signed with + in the table and the last row indicates the number of problems the algorithm is the best. In terms of the best results, ABC variants (ABCV5, ABCV6, ABCV7) that use DE operator in employed bee and onlooker bee phases seem to be superior over the other versions.

In order to compare the algorithms, ANOVA statistical test is conducted to determine whether there is variation within or among different variants of ABC algorithm. If ANOVA test responds as there is significant difference, multiple comparisons are performed to find out which variants differ from the control algorithm, ABCV1. α value to decide an algorithm is significantly different was 0.05. F and P values produced by ANOVA are presented in Table 13. In Table 13, if the variant is significantly better than the control algorithm based on ANOVA and multiple comparisons, it is indicated by (+), else if it is significantly worse than the control algorithm, it is indicated by (−), otherwise it is reported as none. Based on ANOVA test on g01, g04, g08, g11 and g12, there is no significant difference between the variants. On g02, ABCV7 is significantly worse than the control algorithm. On g03, ABCV4 and ABCV9 are worse than ABCV1. On g05 problem, ABCV5, ABCV6 and ABCV7 which are proposed based on DE operators, are worse than ABCV1. On g06, ABCV4 and ABCV8 perform worse compared to the control algorithm. On g07, ABCV2, ABCV5, ABCV6, ABCV7 and ABCV9 differ from the control algorithm ABCV1. Among them, ABCV2 and ABCV9 are worse than ABCV1 while ABCV5, ABCV6 and ABCV7 are better than ABCV1. Similarly, on g09, ABCV5, ABCV6 and ABCV7 are better than ABCV1 while ABCV2 and ABCV4 are worse than ABCV1. On g10 problem ABCV1 shows better performance compared to ABCV2 while ABCV5, ABCV6 and ABCV7 produc better results compared to ABCV1. On g13 problem, ABCV2, ABCV4, ABCV5, ABCV6 and ABCV7 variants are better than ABCV1.

Based on these observations, it can be noted that ABCV5, ABCV6 and ABCV7 retain stability and robustness of the basic ABC algorithm and have shown the efficiency of the DE operator. It should be noted that ABC framework and basic operator has an important effect on the stability. The most stable versions are ABCV6, ABCV2 and ABCV1, respectively. Changing neighborhood topology of the basic ABC operator or making the scouts adaptive based on the fitness does not have significant effect on the performance of the algorithm.

In terms of computational cost, because ABCV2 changes the parameters based on the information each one chosen from a different neighbor, its computational cost is slightly higher than ABCV1. In each operation of ABCV3, a distance matrix is calculated based on the distances between all solution pairs and average distance is computed from the matrix. Therefore computational cost of ABCV3 is higher than ABCV1

Table 11. The results of ABCV9 variant

Problem	Optimum	Best	Mean	Worst	StdDev
g01	-15.000	-15.000	-15.000	-15.000	0.000
g02	0.803619	0.803600	0.792384	0.762961	0.008211
g03	1.000	1.000	0.983335	0.854720	0.034182
g04	-30665.539	-30665.539	-30665.539	-30665.539	0.000
g05	5126.498	5126.485	5185.711	5563.902	86.864
g06	-6961.814	-6961.814	-6961.814	-6961.814	0.000
g07	24.306	24.337	24.517	24.986	0.164383
g08	0.095825	0.095825	0.095825	0.095825	0.000
g09	680.63	680.633	680.640	680.651	0.004974
g10	7049.25	7060.585	7247.809	7729.255	151.192
g11	0.75	0.75	0.75	0.75	0.000
g12	1.000	1.000	1.000	1.000	0.000
g13	0.053950	0.754425	0.952718	0.999837	0.064039

Table 12. An overall comparison of the algorithms in terms of the best results

Problem	ABCv1	ABCv2	ABCv3	ABCv4	ABCv5	ABCv6	ABCv7	ABCv8	ABCv9
g01	+	+	+	+	+	+	+	+	+
g02					+	+			
g03	+	+	+	+	+	+	+	+	+
g04	+	+	+	+	+	+	+	+	+
g05	+	+	+	+	+	+	+		+
g06	+	+	+		+	+	+		+
g07					+	+	+		
g08	+	+	+	+	+	+	+	+	+
g09					+	+	+		
g10					+	+	+	+	
g11	+	+	+	+	+	+	+	+	+
g12	+	+	+	+	+	+	+	+	+
g13							+		
Total	8	8	8	7	12	12	12	7	8

Table 13. ANOVA Table and multiple comparisons to evaluate the results statistically. Control Algorithm is ABCV1 and α value is 0.05.

Problem	F	P	Significantly different groups
g01	1.9602	0.0518	None
g02	178.262	3.9918e-10	ABCV7 (-)
g03	7.4307	6.4332e-009	ABCV4 (-), ABCV9(-)
g04	NaN	NaN	None
g05	8.501	2.8056e-010	ABCV5 (-), ABCV6 (-), ABCV7 (-)
g06	127.985	9.1198e-086	ABCV4 (-), ABCV8(-)
g07	23.987	1.7659e-027	ABCV2(-), ABCV5 (+), ABCV6 (+), ABCV7 (+), ABCV9 (+)
g08	NaN	NaN	None
g09	36.4943	1.6439e-038	ABCV2 (-), ABCV4 (-), ABCV5 (+), ABCV6 (+), ABCV7 (+)
g10	21.2219	9.8283e-02	ABCV2 (-), ABCV5 (+), ABCV6 (+), ABCV7 (+)
g11	1.9087	0.0590	None
g12	NaN	NaN	None
g13	201.1567	6.0648e-107	ABCV2 (+), ABCV4 (+), ABCV5 (+), ABCV6 (+), ABCV7 (+)

and ABCV2. In onlooker phase of ABCV4, in order to produce a mutant solution, the best solution in a neighborhood topology which is constructed based on distances of the solutions and a radius is employed. Its computational cost is close to ABCV3 but higher than ABCV1 and

ABCV2. ABCV5, ABCV6 and ABCV7 do not employ additional procedure compared to the basic algorithm but only replaces the new solution production equation. Hence, their computational cost is close to the basic algorithm. ABCV8 uses the best solution information in its solution production. Since the best solution information is stored in a variable in all variants, it does not introduce more computational cost. In ABCV9, the counters of the solutions are divided by their probability to leave better solutions in the populations for more iterations. It is achieved by only one division operation, so it is computational cost can be assumed to be the same with the ABCV1.

The ABC algorithm is a simple optimization algorithm that can be used for constrained optimization without requiring a priori knowledge. Another advantage of ABC algorithm is that it considers both feasible solutions and infeasible solutions in the population and this provides diversity in the population. In this study, for all variants, the selection strategy is kept as simple as possible and Deb's rules are employed as constraint handling method instead of the greedy selection proposed for unconstrained optimization. However, the performance would change with a different constraint handling method. Analyzing the effect of selection strategies remains as a future work.

5. Conclusion

In this study, ABC algorithm originally proposed for unconstrained optimization has been analyzed on constrained optimization. Different variants of the algorithm have been proposed and compared in terms of efficiency and stability. Depending on the results when DE operators were integrated into ABC algorithm's onlooker phase and the employed bee phase was retained as in ABC algorithm, an improvement in the performance was gained in terms of the best solution and stability. The food source population of ABC algorithm can have both feasible solutions and infeasible solutions, so this provides diversity in the population.

Acknowledgments

This study is supported by Erciyes University, Scientific Research Projects Unit under contract number FBA-10-2959.

References

[1] Goldberg, D. E. . *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[2] C., C. A. C. . A survey of constraint handling techniques used with evolutionary algorithms. Technical report, Laboratorio Nacional de Informtica Avanzada, 1999.

[3] Holland, J. H. . *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[4] Storn, R. and Price, K. . Tr-95-01: Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, Berkeley, CA,, 1995.

[5] M., D. , V., M. , and A., C. . Tr 91-016: Positive feedback as a search strategy. Technical report, Politecnico di Milano, Italy, 1991.

[6] Kennedy, J. and Eberhart, R. C. . Particle swarm optimization. In *1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948", 1995.

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

[8] Koziel, S. and Michalewicz, Z. . Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evol. Comput.*, 7(1):19–44, 1999.

[9] Karaboga, D. and Basturk, B. . *Foundations of Fuzzy Logic and Soft Computing: 12th International Fuzzy Systems Association World Congress, IFSA 2007, Cancun, Mexico, June 18-21, 2007. Proceedings*, chapter Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, pages 789–798. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

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

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

[12] Karaboga, D. and Akay, B. . A survey: Algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1):68–55, 2009.

[13] Karaboga, D. and Gorkemli, B. . A quick artificial bee colony (qabc) algorithm and its performance on optimization problems. *Applied Soft Computing*, 23:227 – 238, 2014.

[14] Akay, B. and Karaboga, D. . A survey on the applications of artificial bee colony in signal, image, and video processing. *Signal, Image and Video Processing*, 9(4):967–990, 2015.

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

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

[17] Mezura-Montes, E. and Coello Coello, C. . 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/.

Bahriye Akay *completed M.Sc. and Ph.D degrees in Erciyes University, Department of Computer Engineering in 2005 and 2009, respectively and performed post-doctoral studies in the University of Birmingham, UK. Her research areas include evolutionary optimization, swarm intelligence and software engineering. She has several papers related to optimization algorithms and their applications. She has been working in Erciyes University as an associated professor since 2013.*

Dervis Karaboga *received the B.Sc. degree in 1983 from the Department of Electronics Engineering, Erciyes University, Turkey and the M.Sc. degree in 1988 from the Department of Electronics and Communication Engineering, Istanbul Technical University, Turkey, and the Ph.D degree in 1994 from Systems Engineering Department, University of Wales, College of Cardiff, UK. He is currently a Professor at the Department of Computer Engineering, Erciyes University, Turkey. His research areas include optimization, fuzzy systems, neural networks, engineering applications of intelligent methods. He has two books on intelligent optimization techniques: The first one is a coauthored book (D.T. Pham and D. Karaboga, Intelligent Optimisation Techniques, Springer-Verlag, London, Surrey, UK, 2000.) and the second one (D. Karaboga, Artificial Intelligence Optimization Algorithms, Nobel Publisher, Ankara, 2004.) is in Turkish. He has several articles published in the journals and papers presented at the conferences related with the intelligent optimization methods and their applications.*

Appendix

g01: *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

$$\begin{aligned}
g_1(\vec{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0\\
g_2(\vec{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0\\
g_3(\vec{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0\\
g_4(\vec{x}) &= -8x_1 + x_{10} \le 0\\
g_5(\vec{x}) &= -8x_2 + x_{11} \le 0\\
g_6(\vec{x}) &= -8x_3 + x_{12} \le 0\\
g_7(\vec{x}) &= -2x_4 - x_5 + x_{10} \le 0\\
g_8(\vec{x}) &= -2x_6 - x_7 + x_{11} \le 0\\
g_9(\vec{x}) &= -2x_8 - x_9 + x_{12} \le 0
\end{aligned}$$

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.

g02: *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

$$\begin{aligned}
g_1(\vec{x}) &= 0.75 - \prod_{i=1}^{n} x_i \le 0\\
g_2(\vec{x}) &= \sum_{i=1}^{n} x_i - 7.5n \le 0
\end{aligned}$$

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 $(g_1 = -10^{-8})$

g03: *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 $f(x^*) = 1$

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

subject to

$$\begin{aligned}
g_1(\vec{x}) &= 85.334407 + 0.0056858x_2x_5\\
&\quad +0.0006262x_1x_4 - 0.0022053x_3x_5\\
&\quad -92 \le 0\\
g_2(\vec{x}) &= -85.334407 - 0.0056858x_2x_5\\
&\quad -0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0\\
g_3(\vec{x}) &= 80.51249 + 0.0071317x_2x_5\\
&\quad +0.0029955x_1x_2 - 0.0021813x_3^2\\
&\quad -110 \le 0\\
g_4(\vec{x}) &= -80.51249 - 0.0071317x2x5\\
&\quad +0.0029955x_1x_2 - 0.0021813x_3^2\\
&\quad +90 \le 0\\
g_5(\vec{x}) &= 9.300961 - 0.0047026x_3x_5\\
&\quad -0.0012547x_1x_3 - 0.0019085x_3x_4\\
&\quad -25 \le 0\\
g_6(\vec{x}) &= -9.300961 - 0.0047026x_3x_5\\
&\quad -0.0012547x_1x_3 - 0.0019085x_3x_4\\
&\quad +20 \le 0
\end{aligned}$$

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

$$\begin{aligned}
&g_1(\vec{x}) = -x4 + x3 - 0.55 \leq 0\\
&g_2(\vec{x}) = -x3 + x4 - 0.55 \leq 0\\
&h_1(\vec{x}) = 1000\sin(-x_3 - 0.25)\\
&\quad +1000\sin(-x_4 - 0.25) + 894.8\\
&\quad -x_1 = 0\\
&h_2(\vec{x}) = 1000\sin(x_3 - 0.25)\\
&\quad +1000\sin(x_3 - x_4 - 0.25) + 894.8\\
&\quad -x_2 = 0\\
&h_3(\vec{x}) = 1000\sin(x_4 - 0.25)\\
&\quad +1000\sin(x_4 - x_3 - 0.25)\\
&\quad +1294.8 = 0
\end{aligned}$$

where $0 \leq x_1 \leq 1200,\ 0 \leq x_2 \leq 1200, -0.55 \leq x_3 \leq 0.55$, and $-0.55 \leq x_4 \leq 0.55$. 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

$$\begin{aligned}
&g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0\\
&g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0
\end{aligned}$$

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. 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

$$\begin{aligned}
&g_1(\vec{x}) = -105 + 4x1 + 5x2 - 3x7 + 9x8 \leq 0\\
&g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0\\
&g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0\\
&g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x4\\
&\quad -120 \leq 0\\
&g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0\\
&g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5\\
&\quad -6x_6 \leq 0\\
&g_7(\vec{x}) = 0.5(x1 - 8)2 + 2(x2 - 4)2 + 3x_5^2 - x6\\
&\quad -30 \leq 0\\
&g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0
\end{aligned}$$

where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$. 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* $f(\vec{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$

subject to

$$\begin{aligned}
&g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0\\
&g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0
\end{aligned}$$

where $0 \leq x_i \leq 10$ $(i = 1, 2)$. 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

$$\begin{aligned}
&g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x3 + 4x_4^2\\
&\quad +5x5 \leq 0\\
&g_2(\vec{x}) = -282 + 7x_2 + 3x_2 + 10x_3^2 + x_4\\
&\quad -x_5 \leq 0\\
&g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0\\
&g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0
\end{aligned}$$

where $-10 \leq x_i \leq 10$, $(i = 1, \ldots, 7)$.
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

$$\begin{aligned}
&g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0\\
&g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0\\
&g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0\\
&g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1\\
&\quad -83.333333 \leq 0\\
&g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4\\
&\quad -1250x_4 \leq 0\\
&g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0
\end{aligned}$$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000, (i = 2, 3)$, $10 \leq x_i \leq 1000, (i = 4, \ldots, 8)$. 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 $-1 \leq x_1 \leq 1$, $-1 \leq x_2 \leq 1$. The optimum solution is $x^* = (\pm 1/\sqrt{(}2), 1/2)$, where $f(x^*) = 0.75$.

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

subject to

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

where $0 \leq x_i \leq 10$, $(i = 1, 2, 3)$ 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

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 0$$
$$h_2(\vec{x}) = x_2 x_3 - 5 x_4 x_5 = 0$$
$$h_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$$

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