

Elitist Artificial Bee Colony for Constrained Real-Parameter Optimization

Efrén Mezura-Montes *Member, IEEE* and Ramiro Ernesto Velez-Koepfel

Abstract—A novel algorithm to solve constrained real-parameter optimization problems, based on the Artificial Bee Colony algorithm is introduced in this paper. The operators used by the three types of bees (employed, onlooker and scout) are modified in such a way that more diverse and convenient solutions are generated. Furthermore, a dynamic tolerance control mechanism for equality constraints is added to the algorithm in order to facilitate the approach to the feasible region of the search space. Finally, two simple local search operators are applied to the best solution found so far. The algorithm, called Elitist-ABC, is tested on 18 test problems based on the experimental design proposed for the CEC'2010 competition on constrained real-parameter optimization. The results obtained are discussed and some conclusions are drawn.

I. INTRODUCTION

Among Swarm Intelligence algorithms[1], those based on honey bees have gained popularity among researchers and practitioners to solve optimization problems. They are characterized by interesting features such as a decentralized control, self-organization and adaptation [2].

Based on the review of the state-of-the-art in [3], there are three types of honey-bee-based algorithms: (1) those based on the mating behavior, (2) those based on the foraging behavior and (3) those based on the queen bee behavior. However, based on the fact that the last type mainly covers special operators utilized in evolutionary algorithms (EAs) and not a new model at all, this paper will consider only the first two classes of algorithms based on bees.

One of the first works inspired on the mating behavior was published in [4] to solve combinatorial optimization problems. From this work, different approaches have been proposed, most of them focused precisely in combinatorial optimization [3]. However, there are some attempts to use this behavior in continuous search spaces like that proposed in [5] and called the Honey Bee Mating Optimization (HBMO) algorithm. A set of best solutions, called queen bees, in the population are recombined with randomly generated solutions called drones, by using a probabilistic function which also considers the velocity and energy of the queen bee during its mating flight. The model considers the addition of heuristics to improve the newly generated broods.

On the other hand, the foraging behavior has inspired different algorithms to solve real-parameter optimization

problems. In [6] the Bees Algorithm (BA) was proposed, where the bees are located in random solutions and the search is focused on “the elite sites” i.e., the best set of solutions in the population. Therefore, more bees are directed to those best sites. The remaining solutions are visited with a lower frequency. A set of scout bees perform random search to promote diversity in the population. A shrinking mechanism in those set of solutions was added into BA in [7] and called Improved Bees Algorithm (IBA). The Virtual Bee Algorithm (VBA) was proposed in [8]. VBA considers a communication process based on waggle dances from bees located in food sources to other bees located in the neighborhood. One of the most popular algorithms within this class is the Artificial Bee Colony (ABC) initially presented in [9] and further tested in [7], [10]. In [11] an improved version with a novel operator based on the Newtonian Law of Universal Gravitation was proposed to improve the exploration ability of the ABC.

To the best of the authors’ knowledge, from the previous approaches, only HBMO and ABC have been adapted to solve constrained real-parameter optimization problems. In [12], the set of feasibility rules proposed in [13] were added as the selection criteria in the original ABC. A recombination operator plus a different rate of use of the scout behavior (responsible to keep diversity in the population) were included in ABC. Finally, HBMO solved just one two-variable optimization problem by using a penalty function [5].

From the literature review summarized above, it is clear that the bee-based algorithms to deal with constrained real-parameter optimization problems are still scarce. Therefore, this paper presents a modified version of the original ABC algorithm in order to participate in the competition on constrained real-parameter optimization.

The document is organized as follows: Section II states the problem of interest. Section III details the original ABC algorithm, while in Section IV the modified ABC algorithm and its corresponding pseudocode are presented. The experimental design proposed for the competition and the obtained results are covered in Section V. Finally, some conclusions and the future work are shown in Section VI

II. STATEMENT OF THE PROBLEM

The problem tackled in this paper is the constrained real-parameter optimization problem, which, without loss of generality, can be defined as to:

Find x which minimizes

$$f(x) \tag{1}$$

subject to

Efrén Mezura-Montes is with the Laboratorio Nacional de Informática Avanzada (LANIA A.C.), Rébsamen 80, Centro, Xalapa, Veracruz, 91000, MEXICO (email: emezura@lania.mx).

Ramiro-Ernesto Velez-Koepfel is with the Escuela de Ingeniería de Antioquía, Km 2 variante al Aeropuerto José María Córdova, Envigado, COLOMBIA (email:ramiro.velez.koepfel@gmail.com).

$$g_i(x) \leq 0, \quad i = 1, \dots, m \quad (2)$$

$$h_j(x) = 0, \quad j = 1, \dots, p \quad (3)$$

where $x \in \mathbb{R}^n$ is the vector of solutions $x = [x_1, x_2, \dots, x_n]^T$ and each x_i , $i = 1, \dots, n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$ which define the search space \mathcal{S} , \mathcal{F} comprises the set of all solutions which satisfy the constraints of the problems and it is called the feasible region; m is the number of inequality constraints and p is the number of equality constraints. Both, the objective function and the constraints can be linear or nonlinear. To handle equality constraints they are transformed into inequalities constraints as follows: $|h_j(x)| - \epsilon \leq 0$, where ϵ is the tolerance allowed (a very small value).

In a similar way to evolutionary algorithms (EAs) [14], the ABC algorithm in its original version lacks a mechanism to deal with the constraints of an optimization problem [15]. Therefore, there are different constraint-handling mechanisms which can be added to ABC in order to find the feasible global optimum solution.

III. ARTIFICIAL BEE COLONY ALGORITHM

The ABC algorithm is based in the interaction of three types of bees: employed, onlooker and scout bees. The number of employed bees usually is the same to the number of onlooker bees. In contrast to other EAs and swarm intelligence algorithms, the solutions in the ABC are represented by the food sources. The bees act as operators over the food sources trying to find the best one among them. In ABC, the number of employed and onlooker bees is the same, and this number is equal to the number of food sources. An employed bee is assigned to one of the sources. Upon reaching the source, the bee will calculate a new solution i.e., fly to another nearby food source from it and retain the best solution (in a greedy selection). The onlooker bees are allocated to a food source based on their profitability. These onlooker bees use the same operator of the employed bee to generate a new food source. When a source does not improve after a certain number of iterations, it is abandoned and replaced by those found by a scout bee, which involves calculating a new solution at random.

Three parameters must be defined by the user: the number of solutions (food sources) SN , the total number of cycles (iterations) of the algorithm $M CN$ and the number of cycles that a non improved solution will be kept before being replaced by a new solution generated by the scout bee mechanism *limit*. The selection process takes place when the employed bees share the information of their food sources in the hive by waggle dances, emulated as a fitness proportional selection. Two replacement processes are required in ABC:(1) in the greedy selection between the current food source and the new one generated either by an employed or an onlooker bee and (2) when an employed bee abandon a food source which could not be improved

within *limit* cycles and a scout bee generates a new one by using a random process. The original operator used by both, employed and onlooker bees, to generate a new solution v_i^g is in Equation 4:

$$v_{i,j}^g = x_{i,j}^g + \phi_{i,j} \cdot (x_{i,j}^g - x_{k,j}^g) \quad (4)$$

where x_i^g represents the solution in which the bee is located at that moment, x_k^g is a randomly chosen food source (and different from x_i^g), $i \in \{1, 2, \dots, SN\}$, $j \in \{1, 2, \dots, n\}$ and $\phi_{i,j}$ is a random real number within $[-1, 1]$ generated at random for every $i \in \{1, 2, \dots, SN\}$ and every $j \in \{1, 2, \dots, n\}$.

However, for the ABC version proposed in [12] for constrained real-parameter optimization a modified version was employed and it is detailed in Equation 5 where a new parameter $0 \leq MR \leq 1$ was considered.

$$v_{i,j}^g \begin{cases} x_{i,j}^g + \phi_{i,j} \cdot (x_{i,j}^g - x_{k,j}^g) & , \text{if } \text{rand}(0,1) < MR \\ x_{i,j}^g & , \text{otherwise} \end{cases} \quad (5)$$

The detailed pseudocode of the original ABC algorithm is presented in Figure 1.

```

1  BEGIN
2  Initialize the set of food sources  $x_i^0$ ,  $i = 1, \dots, SN$ 
3  Evaluate each  $x_i^0$ ,  $i = 1, \dots, SN$ 
4   $g = 1$ 
5  REPEAT
6    FOR  $i = 1$  TO  $SN$ 
7      Generate  $v_i^g$  with  $x_i^{g-1}$  by using Eq. (4)
8      Evaluate  $v_i^g$ 
9      IF  $v_i^g$  is better than  $x_i^{g-1}$ 
10          $x_i^g = v_i^g$ 
11      ELSE
12          $x_i^g = x_i^{g-1}$ 
13      END IF
14    END FOR
15    FOR  $i = 1$  TO  $SN$ 
16      Select, based on fitness proportional selection
17      food source  $x_i^g$ 
18      Generate  $v_i^g$  with  $x_i^g$  by using Eq. (4)
19      Evaluate  $v_i^g$ 
20      IF  $v_i^g$  is better than  $x_i^g$ 
21          $x_i^g = v_i^g$ 
22      END IF
23    END FOR
24    Generate new food sources at random for those
25    whose limit to be improved has been reached
26    Keep the best solution so far
27     $g = g + 1$ 
28  UNTIL  $g = M CN$ 
29  END

```

Fig. 1. Artificial Bee Colony algorithm.

IV. ELITIST ABC

Based on preliminary observations to the original ABC modified to solve constrained real-parameter optimization

problems [12], it was noticed that the algorithm converges faster to a promising region of the search space, causing, sometimes, premature convergence. Therefore, in order to improve its performance in constrained numerical search spaces, five modifications were proposed. The expected effect of all changes is to slow-down convergence by slowing-down the replacement process of the food sources generated by the onlooker bees, modifying the operator of the employed bee and giving the scout bee more chances to generate good solutions in the neighborhood of the best solution so far. Finally, a dynamic mechanism was added to the tolerance for equality constraints with the intention to facilitate the algorithm to approach the tiny feasible region in this type of problems and a local search mechanism was considered to improve those best solutions found. The details of each modification is explained below.

A. Application of the Employed Bee

Acting on every food source, the employed bee modifies its food source x_i^g by using the modified operator in Equation 6.

$$v_{i,j}^g = x_{i,j}^g + 2\phi_i \cdot (x_{k,j}^g - x_{i,j}^g) \quad (6)$$

where v_i^g is the new food source generated and x_k^g is a randomly chosen food source taken from the current population. The ϕ_i value is within the range $[0, 1]$ and it is multiplied by two in order to increase the stepsize of the operator (useful in wider search spaces). This ϕ_i value is kept fixed for every $j \in [1, 2, \dots, n]$.

B. Onlooker Bee Sampling Promising Regions in the Search Space

Instead of improving those better food sources in the population as in the original ABC, each onlooker bee generates a food source v_i^g near the best food source so far x_B^g and the replacement takes place only if the new food source v_i^g is better than that best solution in the population x_B^g i.e., the selection mechanism is not between the current food source x_i^g and the new one recently generated v_i^g . Instead, it is between the new food source v_i^g and the best so far in the population x_B^g . The modified operator is in Equation 7.

$$v_{i,j}^g = x_{i,j}^g + 2\phi_{i,j} \cdot (x_{i,j}^g - x_{B,j}^g) \quad (7)$$

Unlike the modified employed bee operator in Equation 6, the onlooker bee operator generates a different $\phi_{i,j}$ value, within $[0, 1]$, for each $i \in [1, 2, \dots, SN]$ and $j \in [1, 2, \dots, n]$ aiming to generate a more diverse set of solutions near the best one so far. Finally, this best solution x_B^g is always the same for all onlooker bees i.e., it is replaced until the end of each cycle in the algorithm. From this modification on the onlooker bee, the term Elitist is added to the name of this version of the ABC.

C. Modified Scout Bee Operator

A modified scout mechanism, based on an original proposal utilized in PSO [16] is employed in Elitist-ABC. Instead of generating a random food source (as in the original ABC), the scout bee will generate a new food source v_i^g by using the food source subject to be replaced x_i^g as a base to generate a new search direction biased by the best food source so far x_B^g and a randomly chosen food source x_k^g , as indicated in Equation 8, where the ϕ_i value is generated and fixed per each food source $i \in [1, 2, \dots, SN]$.

$$v_{i,j}^g = x_{i,j}^g + \phi_i \cdot (x_{k,j}^g - x_{i,j}^g) + (1 - \phi_i) \cdot (x_{B,j}^g - x_{i,j}^g) \quad (8)$$

The aim of this modified operator is to increase the capabilities of the algorithm to sample solutions within the range of search defined by the current population.

D. Dynamic Tolerance for Equality Constraints

In order to make easier to the ABC the satisfaction of equality constraints, a dynamic mechanism based on the tolerance value ϵ , initially proposed for Evolution Strategies in [17], was considered in this work. The tolerance value is defined with respect to Equation 9.

$$\epsilon = \begin{cases} 1 & , \text{ if } t = 1 \\ e^{-(\frac{E(t) \cdot dec}{E_f})} & , \text{ if } 1 < t < S \\ 0.0001 & , \text{ if } t \geq S \end{cases} \quad (9)$$

where $E(t)$ is the current number of evaluations performed by E-ABC, E_f is the total number of evaluations to be computed and S is the evaluation number when the user wants ϵ becomes to 0.0001. S has to be lower or equal than $E_f \cdot dec$, which controls the speed to reduce ϵ is given by Equation 10, where 9.21034 is the exponent used to get the tolerance value of 1E-4 i.e., $e^{-9.21034} = 1E-4$:

$$dec = \frac{9.21034 * E_f}{S} \quad (10)$$

The complete pseudocode of the Elitist ABC (E-ABC) is shown in Figure 2, where the input parameters are the number of food sources SN , the initial ϵ value, S (for the dynamic equality constraints tolerance). MCN was not considered a parameter in these experiments because the termination condition was controlled by the FEs requested per dimension in [18]. A similar situation occurs with $limit$ which is calculated with $limit = (FEs - SN) / 2 \cdot (SN)^2$. To deal with the constraints of the problem, the three feasibility rules proposed by Deb [13] are used as criteria to compare solutions. They are the following:

- 1) Between two feasible food sources, the one with the best objective function value is chosen.
- 2) If one food source is feasible and the other one is infeasible, the feasible one is chosen.
- 3) Between two infeasible food sources, the one with the lower sum of constraint violation is chosen.

E. Local Search

In order to achieve better results, two local search processes have been used. The first local search works when 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95% and 97% of *FES* have been reached, and its purpose is to improve the best solution achieved so far by generating a set of 1000 new food sources in its neighborhood. Each one of the new solutions v_i^g is generated as shown in Equation 11

$$v_{i,j}^g = x_{B,j}^g + \phi_i \cdot (\text{rand}[0,1](U_i - L_i) \frac{0.1}{0.5(T)}) \quad (11)$$

This local search process generates a different ϕ_i value, within $[-1,1]$, which is fixed in each $j \in [1,2,\dots,n]$. T is a counter, beginning at 1, that increases in 1 each time this local search works. The other local search process consists on slightly modifying each decision variable $x_{B,j}$, $j \in \{1,2,\dots,n\}$ of the best solution so far, by using a small stepsize ($1E-5$ times the size of the variable interval $U_i - L_i$) until either there is no improvement or 100 cycles had been computed per each variable. This local search works when 45%, 50%, 55%, 80%, 82%, 84%, 86%, 88%, 90%, 91%, 92%, 93%, 94%, 95%, 96%, 97%, 98%, and 99% of *FES* have been reached.

V. RESULTS AND DISCUSSION

The experimental design is based on the request for the competition [18], where 18 scalable test problems are considered. The configuration of the computer utilized was as indicated in Table I.

TABLE I
PC DETAILS USED FOR E-ABC EXPERIMENTS

System:	Windows Vista
RAM:	3 GB
Algorithm	E-ABC
CPU:	2.8 GHz
Prog. Language:	Matlab 7.9.0

The parameters values utilized by the approach are summarized in Table II:

TABLE II
PARAMETER VALUES USED BY E-ABC

SN	75
ϵ	1.0
S	45% of <i>FES</i>
<i>FES</i> ratio to perform local search	30%, 40%, 45%, 50%, 55%, 60%, 70%, 80%, 82%, 84%, 86%, 88%, 90%, 91%, 92%, 94%, 95%, 96%, 97%, 98%, 99%
cycle limit for local search	100
stepsize variation for each variable in local search	$1E - 5(U_i - L_i)$

Table III includes the results for the algorithm complexity.

```

1 BEGIN
2 Initialize the set of food sources  $x_i^0$ ,  $i = 1, \dots, SN$ 
3 Evaluate each  $x_i^0$ ,  $i = 1, \dots, SN$ 
4  $g = 1$ 
5 IF There are equality constraints
6 Initialize  $\epsilon$ 
7 END IF
8 REPEAT
9 IF There are equality constraints
10 Evaluate each  $x_i^0$ ,  $i = 1, \dots, SN$  with the  $\epsilon$  value
11 END IF
12 FOR  $i = 1$  TO  $SN$ 
13 Generate  $v_i^g$  with  $x_i^{g-1}$  by using Eq. (6)
14 Evaluate  $v_i^g$ 
15 IF  $v_i^g$  is better than  $x_i^{g-1}$ 
16  $x_i^g = v_i^g$ 
17 ELSE
18  $x_i^g = x_i^{g-1}$ 
19 END IF
20 END FOR
21 FOR  $i = 1$  TO  $SN$ 
22 Generate  $v_i^g$  with  $x_i^{g-1}$  by using Eq. (7)
23 Evaluate  $v_i^g$ 
24 IF  $v_i^g$  is better than  $x_{B,i}^g$ 
25  $x_{B,i}^g = v_i^g$ 
26 ELSE
27  $x_{B,i}^g = x_{B,i}^{g-1}$ 
28 END IF
29 END FOR
30 Apply the modified flight by the scout bees (Eq. 8) for those
31 solutions whose limit to be improved has been reached
32 Apply local search if the percentage of evaluations
33 has been achieved
34 Keep the best solution so far
35 IF There are equality constraints
36 Decrease  $\epsilon$  by using Eq. (9)
37 END IF
38  $g = g + 1$ 
39 UNTIL  $g = MCN$ 
40 END

```

Fig. 2. Elitist Artificial Bee Colony algorithm (E-ABC)

TABLE III
ALGORITHM COMPLEXITY

T1	T2	(T2-T1)/T1
20.193573	23.219179	0.149830
30.088167	32.522573	0.080909

The results obtained by E-ABC in the 18 test problems are reported in Tables IV, V and VI for 10D and in Tables VII, VIII and IX for 30D. It is important to remark that, even E-ABC uses a dynamic tolerance for equality constraints, the results in the aforementioned Tables are based on the tolerance value handled in the competition ($\epsilon = 1E-4$).

From the results in Tables IV, V and VI it was noted that E-ABC found feasible solutions after 200,000 evaluations in sixteen out of eighteen problems (the exceptions were C04 and C11). Besides, in problems C01, C07, C08, C13, C14 and C15 for 10D, feasible solutions were generated before 10% of the total *FES* was processed. It is worth noticing that for some problems, such as C03, C09, C10, C14 and C15 the standard deviation value is very high, which may suggest some lack of robustness in the algorithm, combined with high values for their objective functions. On the other hand,

in some 10D problems, such as C03 and C12, even the best solution was feasible after the total FEs, the median or even the worst solutions were infeasible. This is an interesting behavior of the approach, because it can maintain infeasible solutions late in the process i.e., some kind of diversity is promoted. With the exception of problems C03, C04, C11 and C12, all runs reported complete feasible populations at the end of the process in the remaining problems.

Regarding the results in Tables VII, VIII and IX for 30D, E-ABC presented a similar behavior with respect to 10D problems. Feasible solutions were found in fifteen of eighteen test problems. Moreover, in a similar way to 10D, in problems C01, C07, C08, C13, C14 and C15 for 30D, feasible solutions were generated before 10% of the total FEs was processed. With the exception of problems C03, C04, C11 and C12, in the remaining problems all runs reported feasible solutions. Finally, the standard deviation values in problems C03, C09, C10, C14 and C15 showed a lack of consistency in the algorithm.

The convergence plots in Figure 3 for four 10D problems show that the algorithm was trapped in a local optima in problems C09 and C10. However, it was able to improve solution in problem C15 and mostly in problem C14. Figure 4 suggests that the algorithm is capable of improving the solutions if more than 200,000 FEs were allowed, mostly in problem C18.

The plots for 30D problems in Figure 5 show similar behaviors as those presented in the 10D problems. However, a slower convergence in problems C15 and C14 was observed. Finally, Figure 6 shows that the solution for problem C18 could be improved if more than 600,000 FEs were computed.

VI. CONCLUSIONS AND FUTURE WORK

A novel version of the ABC algorithm to solve constrained real-parameter optimization problems was introduced. Five modifications were implemented to the original ABC: (1) The operator of the employed bee was modified with a larger stepsize. (2) The onlooker bee operator was modified in such a way that its only goal was to improve the best solution found in the previous cycle by generating a search direction towards the vicinity of that best solution. (3) Instead of just generating a new solution at random, the scout bee operator generated solutions biased by a current solution in the population (chosen at random) and the best solution so far. The goal was to increase the probability of generating a good attractor to avoid local optima. (4) To facilitate the generation of feasible solutions a dynamic tolerance for equality constraints was added to view as feasible some near-feasible solutions. Finally, (5) two simple local search operators were applied to the best solution at certain times to improve it.

The results obtained, based on the experimental design proposed for the competition, showed that E-ABC provided a very consistent approach to the feasible region in 10D and in 30D problems. However, the quality and robustness of the solutions found were affected, mostly in 30D problems

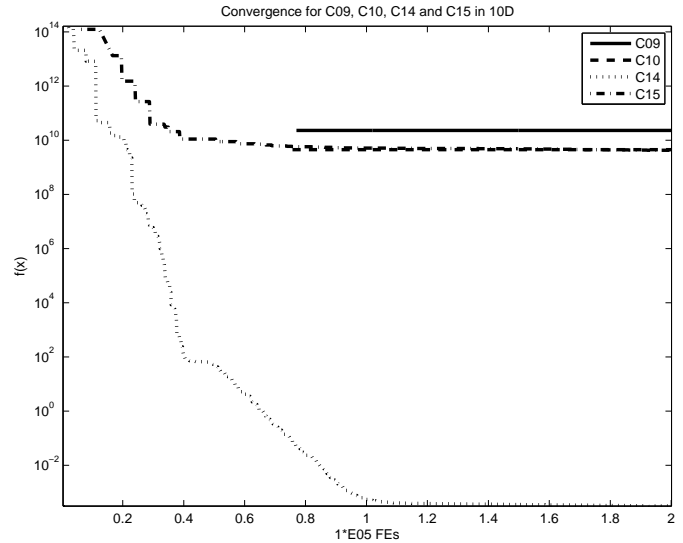


Fig. 3. Convergence plots for problems C09, C10, C14 and C15 for 10D

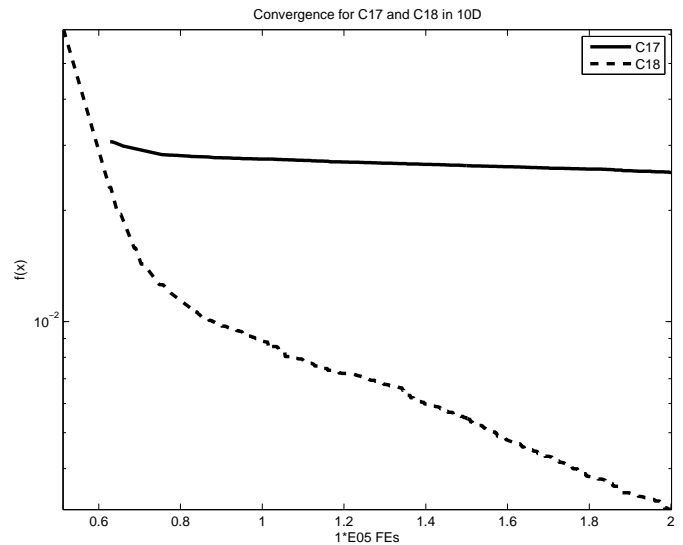


Fig. 4. Convergence plots for problems C17 and C18 for 10D

Finally, for a set of six problems, regardless of dimensionality, E-ABC could find feasible solutions within the first 10% FEs computed and it was able to maintain slightly infeasible solutions in the population, late in the optimization process, for other test problems. These results are still preliminary. Therefore, the future work consists on a sensitivity analysis of E-ABC to its parameters and a mechanism to improve its performance in higher dimensionalities.

ACKNOWLEDGMENTS

The first author acknowledges support from CONACyT through project No. 79809.

REFERENCES

- [1] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.

TABLE IV
FUNCTION VALUES ACHIEVED WHEN FES = $2x10^4$, FES = $1x10^5$, FES = $2x10^5$ FOR 10D PROBLEMS C01-C06

FES		C01	C02	C03	C04	C05	C06
2×10^4	Best	-7.46846E-001	-6.28321E-001 (1)	8.82932E+011 (1)	1.70785E+001 (4)	4.94413E+002 (2)	4.50929E+002 (2)
	Median	-7.24360E-001	-6.91854E-001 (1)	1.06638E+012 (1)	1.77942E+001 (4)	4.46661E+002 (2)	4.34466E+002 (2)
	Worst	-6.65014E-001	-3.70024E-001 (1)	2.49355E+012 (1)	2.14092E+001 (4)	3.98837E+002 (2)	4.99316E+002 (2)
	c	(0,0,0)	(0,1,0)	(1,0,0)	(3,1,0)	(0,2,0)	(0,2,0)
	\bar{v}	0.00000E+000	2.36540E-002	2.56019E+002	2.88234E+001	8.32711E-002	7.01367E-002
	Mean	-7.14960E-001	3.64642E-001	2.65353E+012	1.98888E+001	3.70004E+002	4.40081E+002
	std	2.64416E-002	1.49531E+000	1.05186E+012	1.64264E+000	1.15234E+002	8.56341E+001
1×10^5	Best	-7.47249E-001	-2.15509E+000	8.80208E+011 (1)	2.66172E-001 (4)	4.66347E+001	2.38478E+002
	Median	-7.25563E-001	-9.89099E-001	9.88269E+011 (1)	3.49974E+000 (4)	3.65977E+002	4.48008E+002
	Worst	-6.65028E-001	2.86836E+000	1.91709E+012 (1)	7.63580E+000 (4)	5.36516E+002	5.93442E+002
	c	(0,0,0)	(0,0,0)	(1,0,0)	(1,3,0)	(0,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	1.61027E+000	4.00243E+000	0.00000E+000	0.00000E+000
	Mean	-7.16202E-001	-1.23641E-001	2.45993E+012	4.54092E+000	3.65259E+002	4.38184E+002
	std	2.68619E-002	1.58319E+000	1.01485E+012	3.53090E+000	1.17201E+002	8.59528E+001
2×10^5	Best	-7.47271E-001	-2.15515E+000	8.78821E+011	7.37840E-002 (4)	4.65955E+001	2.38468E+002
	Median	-7.25565E-001	-9.93033E-001	1.40529E+012 (1)	1.44814E-001 (3)	3.65969E+002	4.47997E+002
	Worst	-6.65033E-001	2.86832E+000	2.05672E+012 (1)	2.34519E+000 (4)	5.36501E+002	5.93435E+002
	c	(0,0,0)	(0,0,0)	(0,0,1)	(0,0,3)	(0,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	1.08021E-003	3.58756E-003	0.00000E+000	0.00000E+000
	Mean	-7.16257E-001	-1.24895E-001	2.44507E+012	8.56306E-001	3.65247E+002	4.38168E+002
	std	2.68978E-002	1.58359E+000	1.00967E+012	3.00628E+000	1.17205E+002	8.59536E+001

TABLE V
FUNCTION VALUES ACHIEVED WHEN FES = $2x10^4$, FES = $1x10^5$, FES = $2x10^5$ FOR 10D PROBLEMS C07-C12

FES		C07	C08	C09	C10	C11	C12
2×10^4	Best	3.69381E+001	1.77894E+003	5.68047E+012 (1)	1.05313E+012 (1)	-3.30313E+000 (1)	-4.17379E+002 (1)
	Median	9.66380E+001	1.83068E+004	1.10550E+012 (1)	1.66149E+012 (1)	-2.38642E+000 (1)	4.18727E+001 (1)
	Worst	8.07736E+003	4.66097E+005	8.13694E+012 (1)	5.39585E+012 (1)	-1.30237E+000 (1)	2.15164E+002 (2)
	c	(0,0,0)	(0,0,0)	(0,1,0)	(0,1,0)	(1,0,0)	(1,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	5.76463E-002	8.46351E-002	4.68679E+001	1.13899E+001
	Mean	4.17610E+002	8.67987E+004	3.77802E+012	2.25119E+012	-7.77013E-001	-1.64787E+002
	std	1.59683E+003	1.38159E+005	3.55611E+012	2.75766E+012	3.15500E+000	2.85240E+002
1×10^5	Best	1.34147E-003	4.92935E-001	2.29020E+010	4.49571E+009	1.24410E+000 (1)	-5.70001E+002
	Median	6.45561E+001	1.25149E+002	1.59045E+012	1.16937E+012	-2.53448E+000 (1)	2.27776E+001
	Worst	1.84516E+003	4.64871E+003	7.13143E+012	1.35375E+013	7.57482E+000 (1)	-5.23725E+002 (1)
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	3.42592E+001	0.00000E+000
	Mean	1.46467E+002	4.24139E+002	2.01947E+012	1.74645E+012	-9.43246E-001	-1.80233E+002
	std	3.57511E+002	9.34866E+002	1.81085E+012	2.58327E+012	3.08994E+000	2.73945E+002
2×10^5	Best	1.04468E-003	2.87026E-001	2.28985E+010	4.49358E+009	1.26671E+000 (1)	-5.70036E+002
	Median	6.24215E+001	1.07427E+002	1.59034E+012	1.16922E+012	-3.17814E+000 (1)	1.79467E+001
	Worst	2.74632E+002	4.64274E+003	7.13129E+012	1.35361E+013	7.78918E+000 (1)	-5.32512E+002 (1)
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	3.26889E+001	0.00000E+000
	Mean	7.16094E+001	4.10789E+002	2.01934E+012	1.74620E+012	-1.23017E+000	-1.80089E+002
	std	5.19113E+001	9.35603E+002	1.81083E+012	2.58300E+012	3.03556E+000	2.75758E+002

TABLE VI
FUNCTION VALUES ACHIEVED WHEN FES = $2x10^4$, FES = $1x10^5$, FES = $2x10^5$ FOR 10D PROBLEMS C13-C18

FES		C13	C14	C15	C16	C17	C18
2×10^4	Best	-6.84255E+001	1.15943E+010	1.51870E+012	1.28787E-001 (2)	4.01026E+001 (1)	2.12567E+002 (1)
	Median	-6.36350E+001	1.96386E+012	7.62788E+013	1.34656E-001 (2)	1.99539E-001 (1)	4.51275E+002 (1)
	Worst	-6.17569E+001	1.87242E+013	1.19232E+014 (1)	4.53161E-001 (2)	5.59483E+000 (1)	2.91530E+001 (1)
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,2,0)	(0,1,0)	(0,1,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	3.02142E-002	2.47278E-002	2.63109E-002
	Mean	-6.56544E+001	5.19693E+012	6.41289E+013	2.68285E-001	4.68947E+000	5.17783E+002
	std	2.50709E+000	6.15140E+012	4.41998E+013	1.79063E-001	7.76324E+000	4.96639E+002
1×10^5	Best	-6.84289E+001	5.48224E-004	5.13441E+009	0.00000E+000	2.75391E-002	8.85938E-003
	Median	-6.36350E+001	4.97814E+009	1.44230E+013	6.46348E-002	1.60518E+000	2.67121E+002
	Worst	-6.20674E+001	6.04894E+012	7.33448E+013 (1)	2.69386E-001	3.50137E+001	1.40016E+003
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000
	Mean	-6.56801E+001	7.13830E+011	2.77705E+013	8.49367E-002	3.26571E+000	3.48351E+002
	std	2.50282E+000	1.62014E+012	2.77724E+013	9.19429E-002	6.83695E+000	3.72184E+002
2×10^5	Best	-6.84289E+001	3.14152E-004	4.21611E+009	0.00000E+000	2.53385E-002	3.09847E-003
	Median	-6.36350E+001	2.87761E+003	1.25082E+013	6.15494E-002	1.60308E+000	2.66405E+002
	Worst	-6.20733E+001	9.72178E+011	9.77226E+013	2.67174E-001	3.49734E+001	1.39906E+003
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000
	Mean	-6.56806E+001	8.00410E+010	2.56576E+013	8.34741E-002	3.24290E+000	3.47023E+002
	std	2.50270E+000	2.36608E+011	2.86276E+013	9.10639E-002	6.83136E+000	3.71076E+002

TABLE VII
FUNCTION VALUES ACHIEVED WHEN FES = $6x10^4$, FES = $3x10^5$, FES = $6x10^5$ FOR 30D PROBLEMS C01-C06

FES		C01	C02	C03	C04	C05	C06
$6 * 10^4$	Best	-8.00675E-001	2.95392E+000 (1)	1.07564E+013 (1)	2.56559E+001 (4)	4.20192E+002 (2)	4.86548E+002 (2)
	Median	-7.14645E-001	1.98553E+000 (1)	1.65141E+013 (1)	2.46723E+001 (4)	3.49387E+002 (2)	5.50403E+002 (2)
	Worst	-6.02114E-001	3.79550E+000 (1)	1.39830E+013 (1)	4.64040E+001 (4)	2.83586E+002 (2)	3.29444E+002 (2)
	c	(0,0,0)	(0,1,0)	(1,0,0)	(3,1,0)	(0,2,0)	(0,1,1)
	\bar{v}	0.00000E+000	2.26959E-002	4.82048E+003	8.02910E+001	5.92095E-002	6.51319E-002
	Mean	-7.10174E-001	2.94488E+000	1.18694E+013	2.91521E+001	3.76881E+002	4.79131E+002
$3 * 10^5$	std	4.85621E-002	6.64901E-001	2.36507E+012	6.61648E+000	7.88808E+001	6.32475E+001
	Best	-8.15630E-001	-1.22295E-001	1.05148E+013 (1)	2.42028E+001 (4)	1.46327E+002	3.25983E+002
	Median	-7.39982E-001	2.91551E+000	9.91674E+012 (1)	2.21727E+001 (4)	3.67592E+002	4.91559E+002
	Worst	-6.09260E-001	3.74193E+000	1.17005E+013 (1)	3.96509E+001 (4)	5.01782E+002	5.52704E+002
	c	(0,0,0)	(0,0,0)	(1,0,0)	(3,1,0)	(0,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	4.20466E+001	2.16236E+001	0.00000E+000	0.00000E+000
$6 * 10^5$	Mean	-7.29301E-001	2.57238E+000	1.06929E+013	2.49489E+001	3.71951E+002	4.73908E+002
	std	4.78533E-002	9.43265E-001	2.21555E+012	5.93284E+000	7.88975E+001	6.30249E+001
	Best	-8.16265E-001	-1.23647E-001	1.05323E+013 (1)	2.38950E+001 (3)	1.46311E+002	3.25962E+002
	Median	-7.40123E-001	2.88761E+000	1.12165E+013 (1)	2.18557E+001 (4)	3.67582E+002	4.91517E+002
	Worst	-6.09272E-001	3.73457E+000	9.97238E+012 (1)	3.50045E+001 (4)	5.01709E+002	5.52639E+002
	c	(0,0,0)	(0,0,0)	(0,1,0)	(0,3,1)	(0,0,0)	(0,0,0)
$6 * 10^5$	\bar{v}	0.00000E+000	0.00000E+000	9.20869E-001	3.19424E-001	0.00000E+000	0.00000E+000
	Mean	-7.30554E-001	2.55647E+000	1.07087E+013	2.14572E+001	3.71892E+002	4.73841E+002
	std	4.87589E-002	9.42949E-001	2.20890E+012	6.21913E+000	7.88854E+001	6.30259E+001

TABLE VIII
FUNCTION VALUES ACHIEVED WHEN FES = $6x10^4$, FES = $3x10^5$, FES = $6x10^5$ FOR 30D PROBLEMS C07-C12

FES		C07	C08	C09	C10	C11	C12
$6 * 10^4$	Best	1.77632E+003	7.60564E+007	2.74874E+012 (1)	2.79359E+013 (1)	-1.25833E+000 (1)	4.58268E+002 (1)
	Median	8.67192E+003	2.42604E+008	1.10225E+013 (1)	1.24640E+013 (1)	-5.08906E-001 (1)	-5.23341E+002 (1)
	Worst	3.74392E+004	1.00511E+009	2.32840E+012 (1)	2.54242E+013 (1)	-7.79933E-001 (1)	1.46518E+002 (1)
	c	(0,0,0)	(0,0,0)	(0,1,0)	(0,1,0)	(1,0,0)	(1,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	1.09506E-001	1.08801E-001	1.16450E+004	4.35331E+004
	Mean	9.41492E+003	3.17427E+008	1.51319E+013	1.48739E+013	-4.38030E-001	1.03826E+002
$3 * 10^5$	std	7.43338E+003	2.28998E+008	1.02332E+013	8.34508E+012	5.78025E-001	3.86225E+002
	Best	4.58902E-001	7.17232E-002	2.61407E+012	9.43082E+011	3.02572E-001 (1)	-8.82394E+002
	Median	1.04935E+002	1.65735E+002	1.37699E+013	1.41300E+013	-8.53288E-001 (1)	3.74649E+002
	Worst	1.14829E+003	4.69772E+002	2.20435E+013 (1)	5.37850E+013 (1)	6.62672E-001 (1)	4.12976E+001 (1)
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	3.77738E+001	0.00000E+000
$6 * 10^5$	Mean	1.46691E+002	1.82689E+002	1.60786E+013	1.65852E+013	-6.17349E-001	5.52444E+001
	std	2.12074E+002	1.02474E+002	9.28970E+012	1.24711E+013	6.77934E-001	3.70106E+002
	Best	1.76144E-001	1.60045E-002	2.61374E+012	9.35820E+011	2.77923E-001 (1)	-8.82638E+002
	Median	1.03079E+002	1.41844E+002	1.37613E+013	1.39518E+013	-7.65538E-001 (1)	3.57387E+002
	Worst	1.10835E+003	3.43163E+002	3.98544E+013	3.96332E+013	4.17737E-001 (1)	-3.61922E+001 (1)
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)
$6 * 10^5$	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	3.66579E+001	0.00000E+000
	Mean	1.33290E+002	1.50196E+002	1.60756E+013	1.49864E+013	-5.88529E-001	5.07110E+001
	std	2.05828E+002	7.14877E+001	9.28741E+012	9.77336E+012	6.48625E-001	3.70468E+002

TABLE IX
FUNCTION VALUES ACHIEVED WHEN FES = $6x10^4$, FES = $3x10^5$, FES = $6x10^5$ FOR 30D PROBLEMS C13-C18

FES		C13	C14	C15	C16	C17	C18
$6 * 10^4$	Best	-6.75648E+001	1.38330E+013	7.58647E+013	1.01760E+000 (2)	9.95467E+001 (1)	1.69662E+002 (1)
	Median	-6.52101E+001	4.24217E+013	2.31920E+014	9.93311E-001 (2)	3.39387E+001 (1)	5.00299E+002 (1)
	Worst	-6.23166E+001	6.84703E+013	1.16541E+014 (1)	6.57530E-001 (2)	3.02580E+001 (1)	6.25370E+002 (1)
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,2,0)	(0,1,0)	(0,1,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	2.30836E-002	2.82503E-002	2.60741E-002
	Mean	-6.48263E+001	4.18054E+013	2.89590E+014	9.76353E-001	5.12641E+001	1.48547E+003
$3 * 10^5$	std	1.36188E+000	1.52393E+013	1.96461E+014	1.40954E-001	2.61168E+001	1.78263E+003
	Best	-6.75687E+001	7.19409E+005	2.14404E+013	6.81733E-002	3.36762E+000	2.74819E-001
	Median	-6.52675E+001	4.14575E+011	1.11562E+014	9.23853E-001	2.36020E+001	1.54158E+002
	Worst	-6.23166E+001	4.96544E+012	9.53621E+013 (1)	1.05352E+000	6.18616E+001	1.36340E+003
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000
$6 * 10^5$	Mean	-6.48551E+001	8.60249E+011	1.21509E+014	8.26041E-001	2.73447E+001	3.06638E+002
	std	1.38149E+000	1.33467E+012	7.45325E+013	2.55232E-001	1.66874E+001	3.69264E+002
	Best	-6.75687E+001	3.13491E-001	1.78114E+012	6.25294E-002	3.26752E+000	1.96085E-001
	Median	-6.52689E+001	8.82747E+002	2.93564E+013	9.19836E-001	2.29250E+001	1.48170E+002
	Worst	-6.23166E+001	5.61784E+004	1.50580E+014	1.05256E+000	6.04180E+001	1.32226E+003
	c	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
$6 * 10^5$	\bar{v}	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000	0.00000E+000
	Mean	-6.48559E+001	9.94719E+003	3.78513E+013	8.20722E-001	2.68010E+001	2.93336E+002
	std	1.38213E+000	1.91606E+004	3.44073E+013	2.56988E-001	1.63455E+001	3.52843E+002

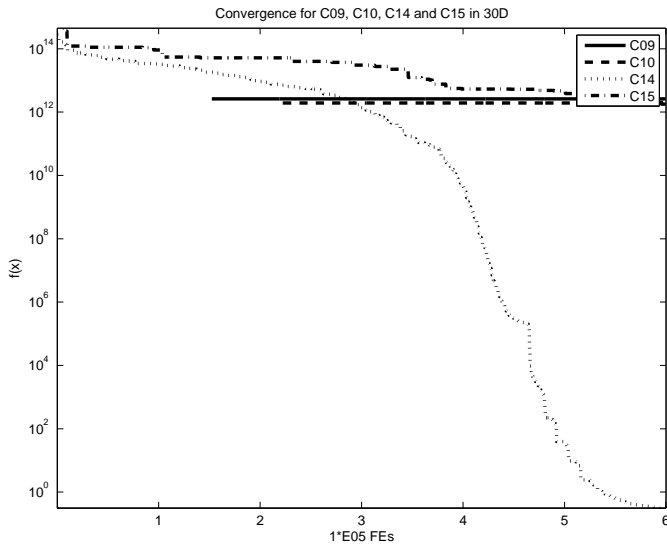


Fig. 5. Convergence plots for problems C09, C10, C14 and C15 for 30D

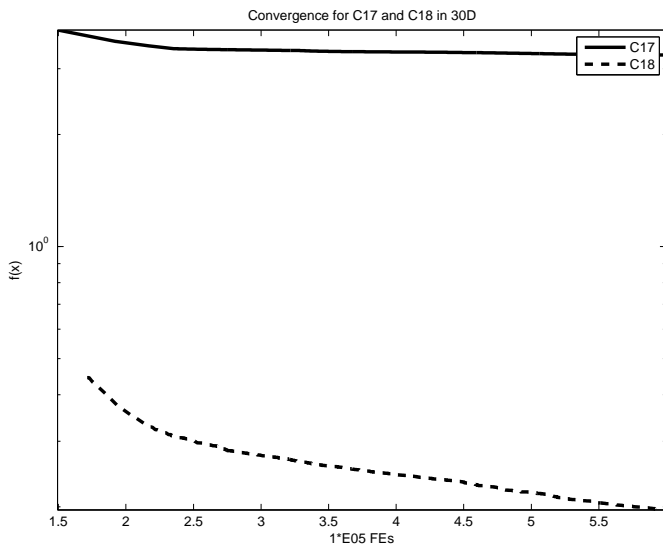


Fig. 6. Convergence plots for problems C17 and C18 for 30D

[2] M. C. Schut, *Scientific Handbook for Simulation of Collective Intelligence*, 2nd ed. Creative Commons, 2007.

[3] A. Baykasoglu, L. Ozbakir, and P. Tapkan, "Artificial bee colony algorithm and its application to generalized assignment problem," in *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, F. T. Chan and M. K. Tiwari, Eds. Vienna, Austria: Itech Education and Pub., 2007, pp. 113–144, ISBN 978-3-902613-09-7.

[4] H. A. Abbass, "Marriage in honey bees optimization (mbo) : A haplometrosis polygynous swarming approach," in *Congress on Evolutionary Computation, CEC 2001*. IEEE Press, 2001, pp. 207–214.

[5] O. B. Haddad, A. Afshar, and M. A. Mario, "Honey-bees mating optimization (hbmo) algorithm: A new heuristic approach for water resources optimization," *Water Resources Management*, vol. 20, pp. 661–680, 2006.

[6] D. Pham, A. Ghanbarzadeh, S. O. E. Ko, S. Rahim, and M. Zaidi, "The bees algorithm: A novel tool for complex optimisation problems," in *2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006)*. Oxford:Elsevier, 2006, pp. 454–461.

[7] D. Karaboga and B. Akay, "Artificial bee colony (abc), harmony search and bees algorithms on numerical optimization," in *Innovative Produc-*

tion Machines and Systems Virtual Conference (IPROMS 2009), 2009, available at: <http://conference.iproms.org/sites/conference.iproms.org/files/IPROMSABCv2.pdf>.

[8] X.-S. Yang, "Engineering optimizations via nature-inspired virtual bee algorithms," in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*. Springer Berlin/Heidelberg, 2005, vol. 3562/2005, pp. 317–323.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Engineering Faculty, Turkey, Tech. Rep., 2005.

[10] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[11] P.-W. Tsai, J.-S. Pan, B.-Y. Liao, and S.-C. Chu, "Enhanced artificial bee colony optimization," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 12, 2009.

[12] D. Karaboga and B. Basturk, "Artificial bee colony(abc) optimization algorithm for solving constrained optimization problems," in *Foundations of Fuzzy Logic and Soft Computing, 12th International Fuzzy Systems Association, World Congress, IFSA 2007*, P. Melin, O. Castillo, L. T. Aguilar, J. Kacprzyk, and W. Pedrycz, Eds. Cancun, Mexico: Springer, Lecture Notes in Artificial Intelligence Vol. 4529, June 2007, pp. 789–798.

[13] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2/4, pp. 311–338, 2000.

[14] A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer Verlag, 2003.

[15] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence. Springer-Verlag, 2009, vol. 198.

[16] H. Lu and W. Chen, "Self-adaptive velocity particle swarm optimization for solving constrained optimization problems," *Journal of Global Optimization*, vol. 41, no. 3, pp. 427–445, July 2008.

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

[18] R. Mallipeddi and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2009.