

# Smart Flight and Dynamic Tolerances in the Artificial Bee Colony for Constrained Optimization

Efrén Mezura-Montes *Member, IEEE*, Mauricio Damián-Araoz and Omar Cetina-Domínguez

**Abstract**—This paper presents an adaptation of a novel algorithm based on the foraging behavior of honey bees to solve constrained numerical optimization problems. The modifications focus on improving the way the feasible region is approached by using a new operator which allows the generation of search directions biased by the best solution so far. Furthermore, two dynamic tolerances applied in the constraint handling mechanism help the algorithm to the generation of feasible solutions. The approach is tested on a set of 24 benchmark problems and its behavior is compared against the original algorithm and with respect to some state-of-the-art algorithms.

## I. INTRODUCTION

The use of heuristic-based search algorithms to solve optimization problems is very common nowadays. Swarm intelligence (SI) comprises a set of search algorithms based on the collaborative behavior observed in simple living organisms such as birds and insects, among others [1]. Besides the two main SI paradigms, ant colony optimization (ACO) [2] and particle swarm optimization (PSO) [3], there are other novel proposals such as the bacterial foraging optimization [4], Fish School Search [5] and glowworm swarm optimization [6]. All of them are characterized by a decentralized control, self-organization and adaptation [7].

Honey bees have been also a source of inspiration for different algorithms to solve optimization problems [8]. There are two main behaviors found in bees which have been modeled as optimization algorithms: (1) The mating behavior [9] and (2) the foraging behavior [10].

The problem tackled in this paper is the constrained nonlinear optimization problem (CNOP), which, without loss of generality, can be defined as to:

Find  $x$  which minimizes

$$f(x) \quad (1)$$

subject to

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

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

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 problem constraints 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:  $|h_j(x)| - \delta \leq 0$ , where  $\delta$  is the tolerance allowed (a very small value).

In a similar way to evolutionary algorithms (EAs) [11], SI algorithms lack a mechanism to deal with the constraints of a CNOP [12]. Therefore, there are different constraint-handling mechanisms which can be added to a SI algorithm in order to bias the search to the feasible region  $\mathcal{F}$  of the search space  $\mathcal{S}$ .

Most of the constraint-handling techniques are usually coupled with EAs [13], [14] and PSO [15], [16], [17]. However, the use and analysis of other SI algorithms to solve CNOPs are still scarce and this is the case of those based on honey bee behaviors.

In this paper the Artificial Bee Colony (ABC), a novel algorithm based on the foraging behavior of honey bees, is adapted with three new mechanisms to improve its capabilities to solve CNOPs. Those new mechanisms are a special operator useful to approach the feasible region from different areas with the aim to improve the quality of feasible solutions. Furthermore, two dynamic tolerances are added to handle constraints (mainly inequality constraints, whose presence is a well-documented source of difficulty in this type of optimization problems [18]).

The document is organized as follows: Section II presents the original ABC algorithm, while in Section III a review of other algorithms based on bee behaviors to solve numerical optimization problems is summarized. After that, the modifications made to the ABC algorithm and the pseudocode of the proposed approach are included in Section IV. The experimental design and the obtained results are shown and discussed in Section V. Finally, some conclusions and the future work are presented in Section VI

## II. ARTIFICIAL BEE COLONY ALGORITHM

Among the algorithms based on the foraging behavior of honey bees, the ABC was designed to deal with numerical optimization problems [10]. ABC is based in two natural processes: The recruitment of bees into a food source and the abandonment of a source.

Efrén Mezura-Montes and Omar Cetina-Domínguez are with the Laboratorio Nacional de Informática Avanzada (LANIA A.C.), Rébsamen 80, Centro, Xalapa, Veracruz, 91000, MEXICO (email: emezura@lania.mx, omarcetina@hotmail.com).

Mauricio Damián Araoz is with the Instituto Tecnológico de Orizaba Department of Computer Systems Engineering Avenida Oriente-9 852, Colonia Emiliano Zapata, Orizaba Veracruz, 94320, MEXICO (email: aroazmd@gmail.com).

An important difference between ABC and other swarm intelligence algorithms is that in the ABC, the solutions of the problem are represented by the food sources, not by the bees. In contrast, bees act as variation operators able to discover (generate) new food sources based on the existing ones. Three types of bees are considered in the ABC: employed, onlooker and scout bees. The number of employed bees is equal to the number of food sources and an employed bee is assigned to one of the sources. Upon reaching the source, the bee will calculate a new solution (fly to another nearby food source) from it and retain the best solution (in a greedy selection). The number of onlooker bees is also the same that the employed bees and they are allocated to a food source based on their profitability. Like the employed bees, they calculate a new solution from its 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. A graphical representation of the model can be found in Figure 1

Three user-defined parameters are required by ABC: the number of solutions (food sources)  $SN$ , the total number of cycles (iterations) of the algorithm  $MCN$  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  $limit$ . ABC uses real-encoding for the solutions of the optimization problem. The selection mechanism used in ABC occurs when the employed bees share the information of their food sources in the hive by waggle dances, emulated as a fitness proportional selection, very similar to a roulette wheel technique used in EAs [19]. The replacement process in ABC takes place in two moments: (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 abandons 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 variation operator used by both, employed and onlooker bees, aims to generate a new candidate solution  $v_{i,j}^g$  at cycle  $g$  by using the formula given in Equation 4:

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

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  $\phi_j$  is a random real number within  $[-1, 1]$  generated at random for every  $j \in \{1, 2, \dots, n\}$ .

The detailed pseudocode of the ABC algorithm is presented in Figure 2. The process begins with the initial set of solutions generated at random and their evaluations (steps 1 – 4 in Figure 2). After that, a cycle that repeats  $MCN$  times starts. Within this cycle there are two inner loops. The first one considers the generation of new solutions by the employed bees by using their assigned food sources (steps 6 – 14 in Figure 2). After that, the second nested loop includes the selection of those better food sources by the onlooker bees and the generation of new solutions (steps 15 – 22 in

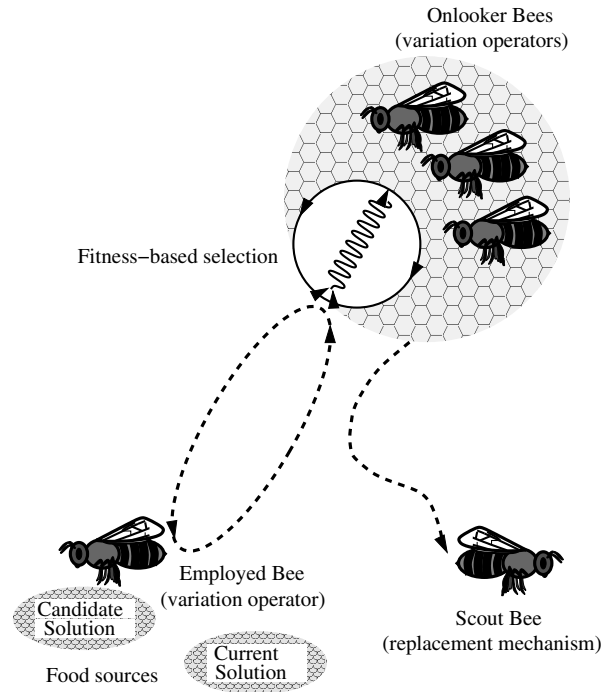


Fig. 1. Graphical representation of the elements in the ABC algorithm.

Figure 2). Finally, the scout bees replace those abandoned solutions by generating new ones (step 23 in Figure 2).

### III. PREVIOUS RELATED WORK

Despite the fact that most of the bee-inspired algorithms have been designed to deal with combinatorial problems [8], there are approaches, besides ABC, focused in numerical search spaces.

Regarding the mating behavior, Bozorg Haddad et al. [20] proposed the Honey Bee Mating Optimization (HBMO) algorithm. A set of queen bees i.e., the best solutions in the population, are recombined with randomly generated solutions i.e., drones, by using a fitness-based probabilistic function which also considers the velocity and energy of the queen bee during its mating flight. Furthermore, worker bees try to improve the newly generated broods. These workers are different heuristic approaches and the use of each worker depends on the success of its improved broods.

The foraging behavior has inspired different algorithms. Pham et al. [21] proposed the Bees Algorithm (BA), where the bees are located in random solutions. Based on the fitness of each solution, the search is focused on the “elite sites” i.e., the best set of solutions in the population. Therefore, more bees (intensive search processes) are directed to those best sites. The remaining solutions are visited with a lower frequency. Finally, a set of remaining bees (scouts) perform random search to promote diversity in the population. In [22], an improved version called Improved Bees Algorithm (IBA) was proposed in order to add a shrinking mechanism to the size of the sets of solutions (called patches).

```

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  $x_l^g$ 
18       Generate  $v_l^g$  with  $x_l^g$  by using Eq. (4)
19       Evaluate  $v_l^g$ 
20       IF  $v_l^g$  is better than  $x_l^g$ 
21         $x_l^g = v_l^g$ 
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  $g = MCN$ 
29 END

```

Fig. 2. Artificial Bee Colony algorithm.

The ABC described in Section II has been tested in different types of unconstrained optimization problems [22], [23]. Furthermore, an improved version called Interactive Artificial Bee Colony (IABC) was proposed in [24]. IABC modifies the operator utilized by the onlooker bee. Such modification was inspired in the Newtonian law of universal gravitation and aims to improve the exploration ability of the ABC.

Xin-She [25] proposed the Virtual Bee Algorithm (VBA) which considers a communication process based on waggle dances based on fitness from bees located in food sources to other bees located in the neighborhood.

From the above mentioned approaches only the ABC has been adapted to solve CNOPs in [26]. The approach kept the same structure ABC has. However, the selection mechanism was modified and three rules based on feasibility, proposed by Deb [27] to be used in an EA, were added to ABC. Furthermore, a recombination mechanism between the original food source and that generated by the reproduction operator in the ABC was included. This operator worked as indicated in Equation 5 where a new parameter  $0 \leq MR \leq 1$  was considered.

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

Finally, the scout bee was activated at certain periods of

time, regardless of the presence of abandoned food sources because the aim was to avoid premature convergence. In [20], there is evidence of the use of HBMO to solve one CNOP but with only two variables and no detail about the constraint-handling utilized (the authors just mention that a penalty function was considered). However, the HBMO required a high number of evaluations to reach competitive results in this problem.

#### IV. SMART FLIGHT ABC

Based on the review of the state-of-the-art on bee-based algorithms to solve CNOPs, summarized in Section III, it is clear that their application in continuous numerical search spaces in presence of constraints is still scarce. Therefore, this paper proposes three modifications to the original ABC algorithm in order to solve CNOPs. These modifications are designed specifically to deal with constrained search spaces. The approach, called Smart Flight ABC, is described below:

##### A. Smart Flight Operator

From observations in the original ABC it was noted that, for CNOPs, it is very difficult for the scout bee to generate, by using a random approach, a solution that could be an attractor to a promising region of the search space. Therefore a modified scout mechanism, based on an original proposal utilized in PSO [28] is adapted now for ABC and called smart flight. Instead of generating a random food source, 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 6.

$$v_{i,j}^g = x_{i,j}^g + \phi \cdot (x_{k,j}^g - x_{i,j}^g) + (1 - \phi) \cdot (x_{B,j}^g - x_{i,j}^g) \quad (6)$$

The aim of the smart flight operator is to increase the capabilities of the algorithm to sample solutions biased to the best solution so far. If this best solution  $x_B^g$  is infeasible, the new solution has a chance to be located near the boundaries of the feasible region. On the other hand, if the best solution is feasible, the smart flight will generate a solution in the area of that promising region.

##### B. Dynamic Tolerances for Constraint-Handling

1)  $\epsilon$ -constrained method: Instead of using the set of rules adopted in the ABC version to solve CNOPs in [26], the proposed ABC utilizes the  $\epsilon$ -constrained method, originally proposed by Takahama and Sakai [29]. The aim of the  $\epsilon$ -constrained method is to transform the original CNOP into an unconstrained optimization problem and uses the sum of constraint violation ( $\Phi$ ) of a given solution (see Equation 7).

$$\Phi(x_k^g) = \sum_{i=1}^m \max(0, g_i(x)) + \sum_{j=1}^p \max(0, |h_j(x)| - \delta) \quad (7)$$

A  $\epsilon$  tolerance allows the comparison between two solutions  $x_i^g$  and  $x_j^g$  by using only their objective function values  $f(x_i^g)$

and  $f(x_j^g)$  when the  $\Phi$  values of both solutions  $\Phi(x_i^g)$  and  $\Phi(x_j^g)$  are within the tolerance value defined by  $\epsilon$  or even when both  $\Phi$  values are the same, as indicated in Equation 8.

$$x_i^g \leq_\epsilon x_j^g \Leftrightarrow \begin{cases} f(x_i^g) \leq f(x_j^g) & , \text{if } \Phi(x_i^g), \Phi(x_j^g) < \epsilon \\ f(x_i^g) \leq f(x_j^g) & , \text{if } \Phi(x_i^g) = \Phi(x_j^g) \\ \Phi(x_i^g) \leq \Phi(x_j^g) & , \text{otherwise} \end{cases} \quad (8)$$

In this way, slightly infeasible solutions will be compared based just in their objective function values, promoting the preservation of those solutions located in promising areas in the boundaries of the feasible region. The  $\epsilon$  value varies with respect to Equation 9.

$$\epsilon(0) = \Phi(x_B^0) \\ \epsilon(g) = \begin{cases} \epsilon(0) \cdot \left(1 - \frac{g}{gc}\right)^{cp} & , \text{if } g < gc \\ 0 & , \text{otherwise} \end{cases} \quad (9)$$

where  $g$  is the cycle number,  $gc$  is the cycle number where the  $\epsilon$  value will become zero and  $cp$  is a parameter to control the speed of reducing the tolerance value. The initial value for  $\epsilon$  corresponds to the sum of constraint violation of the best solution in the initial population.

2) *Tolerance for equality constraints*: In order to make easier to the ABC the satisfaction of equality constraints, a dynamic mechanism based on a tolerance value  $\delta$ , initially proposed for Evolution Strategies in [30], was considered in this work. Recalling from Section I and Equation 7, equality constraints are transformed, by using the  $\delta$  value into inequality constraints. However, in this approach, the value decreases with time until a convenient value i.e.,  $1E-4$  is reached. The formula is detailed in Equation 10

$$\delta(g+1) = \frac{\delta(g)}{dec} \quad (10)$$

The complete pseudocode of the Smart Flight ABC (SF-ABC) is shown in Figure 3, where it is worth remarking that SF-ABC uses binary tournament selection based in Equation 8 instead of fitness proportional selection and that the  $\phi$  value used in Equation 5 is generated once per solution instead of being generated once per variable (as in the original ABC).

## V. RESULTS AND DISCUSSION

In order to test SF-ABC three experiments were carried out: The first one consisted on an indirect comparison with respect to the results reported by the original ABC to solve CNOPs in [26] in 13 well-known benchmark problems. The second test covered a direct comparison in another set of well-known problems (in this case the original ABC was implemented to solve the aforementioned problems). This second set of problems has more instances with equality constraints, which is an important feature to be analyzed in SF-ABC. Finally, a third experiment considered the comparison of the final results of SF-ABC with respect to some state-of-the-art EAs to solve CNOPs. Due to space restrictions, the details of each test problem are not included in this paper.

```

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  $\delta$ 
7 END IF
8 REPEAT
9 Calculate  $\epsilon$  by using Eq. (9)
10 IF There are equality constraints
11 Evaluate each  $x_i^0, i = 1, \dots, SN$  with the  $\delta$  value
12 END IF
13 FOR  $i = 1$  TO  $SN$ 
14 Generate  $v_i^g$  with  $x_i^{g-1}$  by using Eq. (5)
15 Evaluate  $v_i^g$ 
16 IF  $v_i^g$  is better than  $x_i^{g-1}$  (based on Eq. 8)
17  $x_i^g = v_i^g$ 
18 ELSE
19  $x_i^g = x_i^{g-1}$ 
20 END FOR
21 FOR  $i = 1$  TO  $SN$ 
22 Select, based on binary tournament selection
23 food source  $x_l^g$  by using criteria in Eq. (8)
24 Generate  $v_l^g$  with  $x_l^g$  by using Eq. (5)
25 Evaluate  $v_l^g$ 
26 IF  $v_l^g$  is better than  $x_l^g$  (based on Eq. 8)
27  $x_l^g = v_l^g$ 
28 END IF
29 END FOR
30 Apply the smart flight by the scout bees (Eq. 6) for those
31 solutions whose limit to be improved has been reached
32 Keep the best solution so far
33 IF There are equality constraints
34 Decrease  $\delta$  by using Eq. (10)
35 END IF
36  $g = g + 1$ 
37 UNTIL  $g = MCN$ 
38 END

```

Fig. 3. Smart Flight Artificial Bee Colony algorithm (SF-ABC)

However, a summary of the features of each test problem is presented in Table I and the complete definitions can be found in [31].

In the first and third experiments 30 independent runs were computed with SF-ABC. In the second experiment 30 independent runs were computed for both, ABC and SF-ABC. The parameter values used in SF-ABC were the following:  $SN=20$ ,  $MCN=5800$ ,  $limit=MCN/2*SN=145$ ,  $MR=0.8$ ,  $cp=46$ ,  $gc=1160$  (20% of  $MCN$ ),  $\delta=1.0$  and  $dec=1.002$ . The  $MCN$  value was decreased to 3800 in problems with equality constraints because the dynamic tolerance  $\delta$  for those constraints required a re-evaluation of the solutions with the updated tolerance (rows 10 – 12 in Figure 3). Therefore, in order to reach the value of  $1E-4$ , the  $dec$  value was modified to 1.00299 in those problems as well.

The parameters for the original ABC were the same used in SF-ABC:  $SN=20$ ,  $MCN=5800$ ,  $limit=MCN/2*SN=145$ ,  $MR=0.8$ . Both algorithms in experiments 1 and 2 performed the same number of evaluations (240,000 at most). A fixed number cannot be given, because the scout bee activates depending of the behavior of the search and modifies this number.

The comparison of results for the first experiment is

TABLE I

MAIN FEATURES OF TEST PROBLEMS.  $n$  INDICATES THE DIMENSIONALITY OF THE PROBLEM,  $\rho$  IS THE ESTIMATED SIZE OF THE FEASIBLE REGION WITH RESPECT TO THE WHOLE SEARCH SPACE,  $LI$  AND  $NI$  ARE THE NUMBER OF LINEAR AND NONLINEAR INEQUALITY CONSTRAINTS RESPECTIVELY AND  $LE$  AND  $NE$  ARE THE NUMBER OF LINEAR AND NONLINEAR EQUALITY CONSTRAINTS. FINALLY  $A$  INDICATES THE NUMBER OF ACTIVE CONSTRAINTS.

P	n	Function	$\rho$	LI	NI	LE	NE	A
g01	13	quadratic	0.0111%	9	0	0	0	6
g02	20	nonlinear	99.9971%	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230%	0	6	0	0	2
g05	4	cubic	0.0000%	2	0	0	3	3
g06	2	cubic	0.0066%	0	2	0	0	2
g07	10	quadratic	0.0003%	3	5	0	0	6
g08	2	nonlinear	0.8560%	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000%	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g13	5	nonlinear	0.0000%	0	0	0	3	3
g14	10	nonlinear	0.0000%	0	0	3	0	3
g15	3	quadratic	0.0000%	0	0	1	1	2
g16	5	nonlinear	0.0204%	4	34	0	0	4
g17	6	nonlinear	0.0000%	0	0	0	4	4
g18	9	quadratic	0.0000%	0	12	0	0	6
g19	15	nonlinear	33.4761%	0	5	0	0	0
g20	24	linear	0.0000%	0	6	2	12	16
g21	7	linear	0.0000%	0	1	0	5	6
g22	22	linear	0.0000%	0	1	8	11	19
g23	9	linear	0.0000%	0	2	3	1	6
g24	2	linear	79.6556%	0	2	0	0	2

presented in Table II. Based on the statistical results shown, SF-ABC clearly improved the results obtained by the original ABC in problems g05 and g13 (both with equality constraints) and maintained similar competitive results in problems g03 and g11 (both also with equality constraints). SF-ABC also improved the results in problems g06 and g10, both with a very small feasible region with respect to the whole search space. SF-ABC provided similar results with respect to ABC in problems g04, g08 and g12 (these problems are considered easy to solve by most methods of the state-of-the-art) [31]. In problems g07 and g09, the best results found so far was improved by SF-ABC compared with those obtained by ABC. Finally, SF-ABC was affected by premature convergence in problems g01 and g02.

The overall behavior observed in experiment 1 suggests that SF-ABC maintains the competitive performance of ABC, but the new version outperforms the original one in problems with equality constraints and with some type of small feasible regions. However, SF-ABC was affected by a high dimensionality (problem g01 with 13 variables and problem g02 with 20 variables).

The statistical results for the second experiment (which differences were validated by using the Mann-Whitney non-parametric test) are presented in Table III. It is more evident in this second comparison the better performance of SF-ABC, which results outperformed those of the ABC in six problems (g14, g15, g17, g19, g21 and g23) and found better best and mean results in problem g18. ABC was competitive in only two test problems (g16 and g24). Problems g20 and

TABLE II

STATISTICAL RESULTS OF THE INDIRECT COMPARISON BETWEEN ABC AND SF-ABC COVERED IN THE FIRST EXPERIMENT. A RESULT IN BOLDFACE INDICATES A BETTER RESULT.

Function/ Optimal		Algorithms	
		ABC	SF-ABC
<b>g01</b> <b>-15</b>	Best	<b>-15</b>	<b>-15</b>
	Mean	<b>-15</b>	-14.13
	Worst	<b>-15</b>	-12.45
	Std. Dev.	<b>0.00E+00</b>	9.16E-01
<b>g02</b> <b>-0.803619</b>	Best	<b>-0.803598</b>	-0.709034
	Mean	<b>-0.792412</b>	-0.471210
	Worst	<b>-0.749797</b>	-0.319046
	Std. Dev.	<b>1.20E-02</b>	1.05E-01
<b>g03</b> <b>-1</b>	Best	<b>-1.000</b>	<b>-1.000</b>
	Mean	<b>-1.000</b>	<b>-1.000</b>
	Worst	<b>-1.000</b>	<b>-1.000</b>
	Std. Dev.	<b>0.00E+00</b>	<b>2.73E-05</b>
<b>g04</b> <b>-30665.539</b>	Best	<b>-30665.539</b>	<b>-30665.539</b>
	Mean	<b>-30665.539</b>	<b>-30665.539</b>
	Worst	<b>-30665.539</b>	<b>-30665.539</b>
	Std. Dev.	<b>0.00E+00</b>	<b>0.00E+00</b>
<b>g05</b> <b>5126.498</b>	Best	<b>5126.484</b>	5126.49671
	Mean	5185.714	<b>5126.52676</b>
	Worst	5438.387	<b>5126.85967</b>
	Std. Dev.	7.54E+01	<b>7.98E-02</b>
<b>g06</b> <b>-6961.814</b>	Best	<b>-6961.814</b>	<b>-6961.814</b>
	Mean	-6961.813	<b>-6961.814</b>
	Worst	-6961.805	<b>-6961.814</b>
	Std. Dev.	2.00E-03	<b>0.00E+00</b>
<b>g07</b> <b>24.306</b>	Best	24.33	<b>24.3164283</b>
	Mean	<b>24.473</b>	24.6575846
	Worst	<b>25.19</b>	25.5442589
	Std. Dev.	<b>1.86E-01</b>	3.29E-01
<b>g08</b> <b>-0.095825</b>	Best	<b>-0.095825</b>	<b>-0.095825</b>
	Mean	<b>-0.095825</b>	<b>-0.095825</b>
	Worst	<b>-0.095825</b>	<b>-0.095825</b>
	Std. Dev.	<b>0.00E+00</b>	<b>0.00E+00</b>
<b>g09</b> <b>680.63</b>	Best	680.634	<b>680.630107</b>
	Mean	<b>680.640</b>	680.643618
	Worst	<b>680.653</b>	680.857264
	Std. Dev.	<b>4.00E-03</b>	4.09E-02
<b>g10</b> <b>7049.248</b>	Best	7053.90400	<b>7049.54755</b>
	Mean	7224.40700	<b>7116.93411</b>
	Worst	7604.13200	<b>7362.63960</b>
	Std. Dev.	1.34E+02	<b>8.21E+01</b>
<b>g11</b> <b>0.75</b>	Best	<b>0.75</b>	<b>0.75</b>
	Mean	<b>0.75</b>	<b>0.75</b>
	Worst	<b>0.75</b>	<b>0.75</b>
	Std. Dev.	<b>0.00E+00</b>	<b>0.00E+00</b>
<b>g12</b> <b>-1</b>	Best	<b>-1</b>	<b>-1</b>
	Mean	<b>-1</b>	<b>-1</b>
	Worst	<b>-1</b>	<b>-1</b>
	Std. Dev.	<b>0.00E+00</b>	<b>0.00E+00</b>
<b>g13</b> <b>0.05395</b>	Best	0.760000	<b>0.053942</b>
	Mean	0.968000	<b>0.263967</b>
	Worst	<b>1.000000</b>	<b>1.000000</b>
	Std. Dev.	<b>5.50E-02</b>	2.36E-01

TABLE III

STATISTICAL RESULTS OF THE DIRECT COMPARISON BETWEEN ABC AND SF-ABC COVERED IN THE SECOND EXPERIMENT. A RESULT IN BOLDFACE INDICATES A BETTER RESULT. “-” MEANS THAT NO FEASIBLE SOLUTIONS WERE FOUND.

Function/ Optimal		Algorithms	
		ABC	SF-ABC
<b>g14</b> <b>-47.764411</b>	Best	-	<b>-46.667240</b>
	Mean	-	<b>-46.468389</b>
	Worst	-	<b>-43.871811</b>
	Std. Dev.	-	<b>5.20E-01</b>
<b>g15</b> <b>961.715172</b>	Best	961.7150530	<b>961.7150223</b>
	Mean	962.4743929	<b>961.7159869</b>
	Worst	964.5401884	<b>961.7203755</b>
	Std. Dev.	1.22E+00	<b>1.59E-03</b>
<b>g16</b> <b>-1.905155</b>	Best	<b>-1.905155</b>	<b>-1.905155</b>
	Mean	<b>-1.905155</b>	<b>-1.905155</b>
	Worst	<b>-1.905155</b>	<b>-1.905155</b>
	Std. Dev.	<b>1.10E-15</b>	1.49E-14
<b>g17</b> <b>8876.98068</b>	Best	8946.969010	<b>8927.597647</b>
	Mean	8952.787484	<b>8928.864635</b>
	Worst	8958.605958	<b>8938.617114</b>
	Std. Dev.	8.23E+00	<b>3.12E+00</b>
<b>g18</b> <b>-0.865735</b>	Best	-0.865671	<b>-0.866025</b>
	Mean	-0.734206	<b>-0.740724</b>
	Worst	<b>-0.668302</b>	-0.500000
	Std. Dev.	<b>8.31E-02</b>	1.40E-01
<b>g19</b> <b>32.655593</b>	Best	34.484378	<b>32.662603</b>
	Mean	36.087771	<b>33.107187</b>
	Worst	37.248933	<b>34.914034</b>
	Std. Dev.	8.12E-01	<b>5.11E-01</b>
<b>g20</b> <b>0.188446</b>	Best	-	-
	Mean	-	-
	Worst	-	-
	Std. Dev.	-	-
<b>g21</b> <b>193.7783493</b>	Best	-	<b>193.724872</b>
	Mean	-	<b>270.758409</b>
	Worst	-	<b>328.751504</b>
	Std. Dev.	-	<b>5.30E+01</b>
<b>g22</b> <b>382.902205</b>	Best	-	-
	Mean	-	-
	Worst	-	-
	Std. Dev.	-	-
<b>g23</b> <b>-400.0025</b>	Best	-	<b>-350.125153</b>
	Mean	-	<b>-121.373529</b>
	Worst	-	<b>276.002998</b>
	Std. Dev.	-	<b>1.58E+02</b>
<b>g24</b> <b>-5.508013</b>	Best	<b>-5.508013</b>	<b>-5.508013</b>
	Mean	<b>-5.508013</b>	<b>-5.508013</b>
	Worst	<b>-5.508013</b>	<b>-5.508013</b>
	Std. Dev.	<b>0.00E+00</b>	<b>0.00E+00</b>

g22 could not be solved by any algorithm compared. Furthermore ABC could not find feasible solutions in problems g14, g21 and g23.

SF-ABC could avoid local optimum solutions in problems with equality constraints (g14, g15, g17, g21 and g23) with very small feasible regions. This behavior suggests that the combination of the smart flight operator coupled with the dynamic tolerances for the sum of constraint violation and the equality constraints improves the approach to the feasible region of the search space by reaching it from more promising directions.

Finally, the comparison against some state-of-the-art algorithms to solve CNOPs is presented in Table IV. Problems g14 to g24 were not considered in this comparison because

no results were found for the approaches reported in the paper. The algorithms used in this comparison were the self-adaptive parameter-free penalty function (SAPF-GA) by Tessema & Yen [32], the multi-objective concepts added to an evolutionary algorithm (HCOEA) by Wang et al. [33] and the trade-off models based on feasibility (ATMES) by Wang et al. [34]. SF-ABC provided very competitive results in nine test problems (g03, g04, g05, g06, g07, g08, g09, g11 and g12). As expected from the results in the first experiment, SF-ABC was not competitive in problems g01 and g02. In problem g10 HCOEA was clearly superior in all statistical results and in problem g13 both HCOEA and ATMES were better than SF-ABC. Finally, the number of evaluations required by SF-ABC is the same computed by HCOEA and ATMES (240,000) and it is lower than that reported by SAPF-GA (500,000).

## VI. CONCLUSIONS AND FUTURE WORK

A novel version of the ABC algorithm to solve CNOPs, called Smart Flight ABC has been presented in this paper. A new operator for the scout bee, called smart flight was designed with the aim to generate search directions towards the neighborhood of the best solution in the current population, instead of just generating new solutions at random. Moreover, the constraint-handling mechanism considered the combination of two dynamic tolerance values related with the sum of constraint violation, based on the  $\epsilon$ -constrained method, and with the presence of equality constraints. The expected effect of these modifications is a more convenient approach to the feasible region in order to find better solutions within it, mostly in problems with a very small feasible region. The results obtained in a set of well-known problems showed that SF-ABC was able to provide better results with respect to ABC in the aforementioned type of problems. However, premature convergence was observed in problems with high dimensionality and larger feasible regions in presence of only inequality constraints. Finally, the overall performance of SF-ABC was similar to those provided by state-of-the-art EAs to solve CNOPs with an also similar or lower number of evaluations computed. A sensitivity analysis of SF-ABC to its parameters and a mechanism to improve the performance in presence of higher dimensionalities are part of the future work in this research. Furthermore, some experiments will be carried out to analyze in which type of problems the smart flight is more efficient. Finally, some other performance measures e.g., feasibility rate and convergence plots, will be utilized to improve the analysis of the behavior observed in the algorithm.

## ACKNOWLEDGMENTS

The first author acknowledges support from CONACyT through project No. 79809. The second author acknowledges support from CONACyT through a scholarship to finish his BSc thesis. The third author acknowledges support from CONACyT through a scholarship to pursue graduate studies at LANIA.

TABLE IV

COMPARISON OF RESULTS OF SF-ABC WITH RESPECT TO STATE OF THE ART ALGORITHMS (SAPF-GA [32], HCOEA [33] AND ATMES [34]) ON 13 BENCHMARK FUNCTIONS (G01-G13). A RESULT IN BOLDFACE INDICATES A BETTER RESULT OR THAT THE GLOBAL OPTIMUM (OR BEST KNOWN SOLUTIONS) WAS REACHED.

Function/ Optimal		Algorithms			
		SAPF-GA [32]	HCOEA [33]	ATMES [34]	SF-ABC
g01 -15	Best	<b>-15</b>	<b>-15</b>	<b>-15</b>	<b>-15</b>
	Mean	-14.552	<b>-15</b>	<b>-15</b>	-14.13
	Worst	-13.097	-14.999998	<b>-15</b>	-12.45
	Std. Dev.	7.00E-01	4.30E-07	<b>1.60E-14</b>	9.16E-01
g02 <b>-0.803619</b>	Best	-0.803202	-0.803241	<b>-0.803388</b>	-0.709034
	Mean	-0.755798	<b>-0.801258</b>	-0.790148	-0.471210
	Worst	-0.745712	<b>-0.792363</b>	-0.756986	-0.319046
	Std. Dev.	1.33E-01	<b>3.83E-03</b>	1.30E-02	1.05E-01
g03 -1	Best	<b>-1</b>	<b>-1</b>	<b>-1</b>	<b>-1</b>
	Mean	-0.964	<b>-1</b>	<b>-1</b>	<b>-1</b>
	Worst	-0.887	<b>-1</b>	<b>-1</b>	<b>-1</b>
	Std. Dev.	3.01E-01	<b>1.30E-12</b>	5.90E-05	2.73E-05
g04 <b>-30665.539</b>	Best	-30665.401	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>
	Mean	-306659.221	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>
	Worst	-30656.471	<b>-30665.539</b>	<b>-30665.539</b>	<b>-30665.539</b>
	Std. Dev.	2.04E+00	5.40E-07	7.40E-12	<b>0.00E00</b>
g05 <b>5126.498</b>	Best	5126.907	5126.4981	5126.498	<b>5126.49671</b>
	Mean	5214.232	<b>5126.4981</b>	5127.648	5126.52676
	Worst	5564.642	<b>5126.4981</b>	5135.256	5126.85967
	Std. Dev.	2.47E+02	<b>1.73E-07</b>	1.80E+00	7.98E-02
g06 <b>-6961.814</b>	Best	-6961.046	<b>-6961.81388</b>	<b>-6961.814</b>	<b>-6961.814</b>
	Mean	-6953.061	<b>-6961.81388</b>	<b>-6961.814</b>	<b>-6961.814</b>
	Worst	-6943.304	<b>-6961.81388</b>	<b>-6961.814</b>	<b>-6961.814</b>
	Std. Dev.	5.88E+00	<b>8.51E-12</b>	<b>4.60E-12</b>	<b>0.00E+00</b>
g07 <b>24.306</b>	Best	24.838	24.3064582	<b>24.306</b>	24.3164283
	Mean	27.328	<b>24.3073989</b>	24.316	24.6575846
	Worst	33.095	<b>24.3092401</b>	24.359	25.5442589
	Std. Dev.	2.17E+00	<b>7.12E-04</b>	1.10E-02	3.29E-01
g08 <b>-0.095825</b>	Best	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>
	Mean	-0.095635	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>
	Worst	-0.092697	<b>-0.095825</b>	<b>-0.095825</b>	<b>-0.095825</b>
	Std. Dev.	1.06E-03	<b>2.42E-17</b>	<b>2.80E-17</b>	<b>0.00E00</b>
g09 <b>680.63</b>	Best	680.773	<b>680.630057</b>	<b>680.63</b>	<b>680.630107</b>
	Mean	681.246	<b>680.630057</b>	680.639	680.643618
	Worst	682.081	<b>680.630058</b>	680.673	680.857264
	Std. Dev.	3.22E-01	<b>9.41E-08</b>	1.00E-02	4.09E-02
g10 <b>7049.248</b>	Best	7069.981	<b>7049.2866</b>	7052.253	7049.54755
	Mean	7238.964	<b>7049.52544</b>	7250.437	7116.93411
	Worst	7489.406	<b>7049.98421</b>	7560.224	7362.63960
	Std. Dev.	1.38E+02	<b>1.50E-01</b>	1.20E+02	8.21E+01
g11 <b>0.75</b>	Best	<b>0.749</b>	<b>0.75</b>	<b>0.75</b>	<b>0.75</b>
	Mean	0.751	<b>0.75</b>	<b>0.75</b>	<b>0.75</b>
	Worst	0.757	<b>0.75</b>	<b>0.75</b>	<b>0.75</b>
	Std. Dev.	2.00E-03	<b>1.55E-12</b>	<b>3.40E-04</b>	<b>0.00E00</b>
g12 -1	Best	<b>-1</b>	<b>-1</b>	<b>-1</b>	<b>-1</b>
	Mean	-0.99994	<b>-1</b>	<b>-1</b>	<b>-1</b>
	Worst	-0.999548	<b>-1</b>	-0.994	<b>-1</b>
	Std. Dev.	1.41E-04	<b>0.00E+00</b>	1.00E-03	<b>0.00E+00</b>
g13 <b>0.05395</b>	Best	<b>0.053941</b>	<b>0.0539498</b>	<b>0.05395</b>	<b>0.053942</b>
	Mean	0.28627	<b>0.0539498</b>	<b>0.053959</b>	0.263967
	Worst	0.885276	<b>0.0539499</b>	<b>0.053999</b>	1
	Std. Dev.	2.75E-01	<b>8.68E-08</b>	<b>1.30E-05</b>	2.36E-01

## REFERENCES

- [1] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.
- [2] M. Dorigo, V. Maniezzo, and A. Colnori, "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man and Cybernetics-Part B*, vol. 26, no. 1, pp. 29–41, 1996.
- [3] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. UK: Morgan Kaufmann, 2001.
- [4] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.
- [5] C. J.A.Bastos-Filho, F. B.de-Lima-Neto, A. J.C.C.Lins, A. I.S.Nascimento, and M. P.Lima, "Fish school search," in *Nature-Inspired Algorithms for Optimisation*, R. Chiong, Ed. Heidelberg, Germany: Springer-Verlag, 2009, pp. 261–277.
- [6] K. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal optimization functions with collective robotics applications," *Multiagent and Grid Systems*, vol. 2, no. 3, pp. 209–222, 2006.
- [7] M. C. Schut, *Scientific Handbook for Simulation of Collective Intelligence*, 2nd ed. Creative Commons, 2007.
- [8] 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.
- [9] 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.
- [10] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Engineering Faculty, Turkey, Tech. Rep., 2005.
- [11] A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer Verlag, 2003.
- [12] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence. Springer-Verlag, 2009, vol. 198.
- [13] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.
- [14] C. A. Coello Coello, "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*, vol. 191, no. 11–12, pp. 1245–1287, January 2002.
- [15] C. K. Monson and K. D. Seppi, "Linear equality constraints and homomorphous mappings in PSO," in *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, vol. 1. Edinburgh, Scotland: IEEE Service Center, September 2005, pp. 73–80.
- [16] L. Cagnina, S. Esquivel, and C. Coello-Coello, "A bi-population PSO with a shake-mechanism for solving constrained numerical optimization," in *2007 IEEE Congress on Evolutionary Computation (CEC'2007)*. Singapore: IEEE Press, September 2007, pp. 670–676.
- [17] E. Mezura-Montes and J. I. Flores-Mendoza, "Improved particle swarm optimization in constrained numerical search spaces," in *Nature-Inspired Algorithms for Optimization*, R. Chiong, Ed. Springer-Verlag, Studies in Computational Intelligence Series, 2009, ISBN: 978-3-540-72963-1., 2009, vol. 193, pp. 299–332.
- [18] E. Mezura-Montes and C. A. Coello Coello, "Identifying on-line behavior and some sources of difficulty in two competitive approaches for constrained optimization," in *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, vol. 2. Edinburgh, Scotland: IEEE Press, September 2005, pp. 1477–1484.
- [19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley Publishing Co., 1989.
- [20] 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.
- [21] 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.
- [22] D. Karaboga and B. Akay, "Artificial bee colony (abc), harmony search and bees algorithms on numerical optimization," in *Innovative Production Machines and Systems Virtual Conference (IPROMS 2009)*, 2009, available at: <http://conference.iproms.org/sites/conference.iproms.org/files/IPROMSABCv2.pdf>.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] T. Takahama and S. Sakai, "Constrained optimization by the  $\epsilon$  constrained differential evolution with gradient-based mutation and feasible elites," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*. Vancouver, BC, Canada: IEEE, July 2006, pp. 308–315.
- [30] 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.
- [31] J. J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. A. Coello Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., December, 2005, available at: <http://www.lania.mx/~emezura>.
- [32] B. Tessema and G. G. Yen, "A self adaptative penalty function based algorithm for constrained optimization," in *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*. Vancouver, BC, Canada: IEEE, July 2006, pp. 950–957.
- [33] Y. Wang, Z. Cai, G. Guo, and Y. Zhou, "Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems," *IEEE Transactions on Systems, Man and Cybernetics Part B–Cybernetics*, vol. 37, no. 3, pp. 560–575, June 2007.
- [34] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 80–92, February 2008.