Improved Swarm Bee Algorithm for Global Optimization

Tarun Kumar Sharma
Department of Paper Technology
Indian Institute of Technology
Roorkee, India

Millie Pant
Department of Paper Technology
Indian Institute of Technology
Roorkee, India

ABSTRACT
Artificial Bee Colony (ABC) algorithm simulates the foraging behavior of honey bee colonies. ABC is an optimization technique, which is used in finding the best solution from all feasible solutions. However, there is still an insufficiency in ABC regarding improvement in exploitation and convergence speed. In order to improve the performance of ABC we embedded PSO into ABC. As PSO has memory, knowledge of good solutions is retained by all the particles. In addition, to improve the convergence speed, the initial population of food sources is produced using the union of random generated population using random numbers and chaotic systems. This modification in basic ABC results in new search mechanism, ISBC (Improved Scout Bee Colony). Experiments are conducted on a set of 6 shifted benchmark functions. The results demonstrate good performance of ISBC in solving complex numerical optimization problems when compared with two ABC-based algorithms.

General Terms
Evolutionary Algorithms, Optimization, Performance of ABC, Initial Food Locations in ABC.

Keywords
Artificial Bee Colony, Convergence, Exploration and Exploitation, Chaotic Maps.

1. INTRODUCTION
ABC, inspired by the intelligent foraging behavior of honeybees, was proposed by Karaboga in 2005 [1]. It's gradually increasing interest of many researchers because of its simplicity, outstanding performance, wide applicability and fewer control parameters. The basic ABC has been compared with other population based evolutionary algorithms, such as GA, PSO, and DE [2][3]. ABC has been used to solve real-world problems such as flow shop scheduling problems [4], minimum spanning tree problems [5], and parameter estimation problems [6].However like other evolutionary algorithms ABC also faces some limitations while handling complex multimodal functions, functions having narrow curve valley and its stochastic nature slows down the convergence speed [7]. The reason behind all this is improper balance between exploration and exploitation. ABC is good at exploration but poor at exploitation [8].NM simplex search has been introduced into ABC, and a hybrid simplex ABC was proposed for inverse analysis to improve the performance of ABC [9].

In this paper we combined ABC & PSO [10-11] global optimization algorithms, and propose the novel hybrid algorithm ISBC. As PSO has memory, knowledge of good solutions is retained by all the particles. Moreover, PSO & ABC both work with an initial population of solutions. So combining the searching abilities of both algorithms seems to be a reasonable approach to improve the convergence speed and to balance exploration & exploitation of basic ABC.

The paper is organized as follows. Section 2 provides the compact reviews of the basic ABC. Chaotic system is presented in Section 3. Section 4 discusses the proposed algorithm. In section 5 experimental settings & performance criterion is defined. The proposed algorithm is tested against a set of 6 shifted benchmark test functions and the results are compared in section 6. Finally, Section 7 summarizes the contribution of this paper along with some future research directions.

2. ARTIFICIAL BEE COLONY
The artificial bee colony classifies the foraging artificial bees into three groups of bees: employed bees, onlookers and scouts. The first half of the colony consists of the employed artificial bees and the second half includes the onlookers. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source has been exhausted by the bees becomes a scout. The pseudo-code of the algorithm is given in Fig. 1.Step-by-step procedure of ABC is given as follows:

(a) Randomly distributed food sources are generated over a D-dimensional search space.

(b) An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, P_i, calculated by the expression:

$$P_i = \frac{fit_i}{\sum_{i=1}^{NP} fit_i} \tag{1}$$

Where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and NP is the number of food sources which is equal to the number of employed bees. In order to produce a candidate food position from the old one in memory, the ABC uses the following expression:

$$V_{ij} = x_{ij} + r_{ij}(x_{ij} - x_{kj}) \tag{2}$$

Where $k \in \{1, 2, \ldots, NP\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indexes. Further, $k \neq i$ and r_{ij} is a random number between [-1, 1].

(c) In ABC algorithm, "*limit*" is an important control parameter, it controls the times of updates of a certain solution. If a solution cannot be improved further through a predetermined number of cycles called limit then that solution is assumed to be abandoned, and the employed bee becomes a

scout. If the solution is x_i and $j \in \{1, 2, \ldots, D\}$ to be abandoned, then a new solution produced randomly would replace x_i by:

$$x_{ij} = x^j_{\min} + rand(0,1)(x^j_{\max} - x^j_{\min}) \quad (3)$$

where $x^j_{\min}$ is the lower bound of the parameter *j* and $x^j_{\max}$ is the upper bound of the parameter *j*.

Begin
1. Initialize the population of food sources x_i, $i = 1,\ldots,SN$
2. Evaluate each food source x_i, $i = 1,\ldots,SN$
3. *cycle* = 1

Repeat
For each food source x_i in the population
4. Generate a new food source v_i by its corresponding employed bee (*Eq. 2*)
5. Evaluate v_i
6. Keep the best solution between x_i and v_i

End
7. Select, based on fitness proportional selection, the food sources to be visited by onlooker bees

For each food source x_i chosen by an onlooker bee
8. Generate a new food source v_i by its corresponding onlooker bee (*Eq. 2*)
9. Evaluate v_i
10. Keep the best solution between x_i and v_i

End
11. Use the scout bee to replace those abandoned food sources
12. Save in memory the best food source so far
13. *cycle* = *cycle* + 1

Until *cycle*
End

Fig 1: Pseudocode of ABC

3. CHAOTIC MAPS

Most of the chaotic maps available in the literature possess certainty, ergodicity and the stochastic property. They have also been used together with some heuristic optimization algorithms [12][13] to express optimization variables. The choice of chaotic sequences is justified theoretically by their unpredictability, i.e., by their spread-spectrum characteristic, non-periodic, complex temporal behavior, and ergodic properties. A chaotic map is a discrete-time dynamical system

$$X_{k+1} = f(X_k), \qquad 0 < X_n < 1, \quad k = 0,1,2,\ldots$$

running in chaotic state. The chaotic sequence $\{X_k : k = 0,1,2,..\}$ can be used as spread-spectrum sequence like a random number sequence. Chaotic sequences have been proven easy and fast to generate and store, there is no need for storage of long sequences [14]. Merely a few functions (chaotic maps) and few parameters (initial conditions) are needed even for very long sequences. In addition, an enormous number of different sequences can be generated simply by changing its initial condition. Moreover these sequences are deterministic and reproducible. Recently, chaotic sequences have been adopted instead of random sequences and very interesting and somewhat good results have been shown in many applications such as secure transmission [15][16], and nonlinear circuits [17], DNA computing [18], image processing [19].

The spread spectrum property of chaotic sequences can particularly be useful for population based nature inspired algorithms like that of ABC, which depend a lot on generation of random numbers. The selected chaotic maps that produce chaotic numbers in (0, 1) for the experiments have been listed in Table 1.

4. ISBC: Proposed Algorithm

4.1 Initial Population using Chaotic Map

Population initialization is a crucial task in evolutionary algorithms which affects the convergence speed as well as the quality of the final solution. When no information about the solution is available, then random initialization is the commonly used method to generate candidate initial population. The chaotic maps owes the randomness and sensitivity dependence on the initial conditions of chaotic maps, chaotic maps have been used to initialize the population so that the search space information can be extracted to increase the population diversity in [20]. It is intended to enhance the global convergence. This paper proposes a novel initialization approach which employs union of randomly generated uniform random numbers and chaotic systems to generate initial population. In, our case sinusoidal iterator is selected and its equation is defined as follows:

$$ch_{m+1} = \sin(\pi ch_m), ch_m \in (0,1), m = 0,1,2,\ldots,m.$$

where k is the iteration counter and K is the preset maximum number of chaotic iterations. Hence the algorithm given in Fig. 2 is used to generate the initial population.

4.2 Modified Search

Exploration and exploitation are the two important aspects in evolutionary computing paradigms. Exploration is the ability to search the solution space to find promising new solutions, and exploitation is the ability to find the optimum solution in the neighborhood of a good solution. The algorithm is improved to balance between the above aspects. To enrich the searching behavior and to avoid being trapped into local optimum, the memory retaining concept of Particle Swarm is incorporated into the ABC. In PSO, the particles tend to move towards good areas in the search space by the information sharing mechanism. The particle moves to a new position calculated by the velocity updated at each time step t.

Further the scout bee phase of the algorithm uses Eq. (4) i.e. *randomized localization*, instead of using Eq. (3), which increases the convergence speed.

$$X_{ij} = x_{tb} + rand()(x_{r2} - x_{r3}) \quad (4)$$

where x_{r1}, x_{r2}, x_{r3} are randomly three distinct individuals from population and x_{tb} is the best of these individuals.

The Pseudocode of the proposed algorithm is given in Fig. 3.

1. Set the population of Food sources, SN and chaotic iteration N≥100.
2. for i=1 to SN
3. do
4. for j=1 to n
5. do
6. $x^j_i = x^j_{\min} + rand()(x^j_{\max} - x^j_{\min})$
7. end for

8. end for
{Initializing population of Food sources using chaotic systems}
9. for i=1 to SN
10. do
11. for j=1 to n
12. do
13. Randomly initialize variables $ch_{0,j} \in (0,1)$, set iteration counter N=0
14. for m=1 to N do
15. $ch_{m+1,j} = \sin(\pi c ch_{m,j})$
16. end for
17. $x_{ij} = x_{\min j} + ch_{mj}(x_{\max j} - x_{\min j})$
18. end for
19. end for
20. Set the individual counter i=1,j=1
21. Selecting SN fittest individuals from the set of $\{x_{ij} \cup OP_{ij}\}$ as initial population.

Fig 2: Pseudocode to Initialize the Food sources

5. EXPERIMENTAL SETTINGS AND PERFORMANCE CRITERION

5.1 Experimental Settings

The proposed algorithm is tested on 6 shifted benchmark problems given taken from literature [21]. ABC and the proposed variants are implemented on Dev-C++ and the experiments are conducted on a computer with 2.00 GHz Intel (R) core (TM) 2 duo CPU and 2- GB of RAM. For each problem, all the algorithms are independently executed 30 times and the average is recorded. The parameters' setting is taken as follows:

Population size	*100 (i.e SN=50)*
limit	*100*
Value to Reach (VTR)	10^{-15}
Maximum MCN	*8000*
ω	*0.7*
$C_1=C_2$	*1.49*
D (Dimension)	*10, 30*

5.2 Performance Criteria

Mean Fitness, Standard Deviation, and Best: The average of function fitness value that an algorithm can find, using predefined MCN, is recorded in each run and then average of the function fitness values are calculated. Also the average, standard deviation, best and worst of the fitness values are calculated.

MCN: The MCN (Maximum Cycle Number) is recorded when the VTR is reached before to reach maximum MCN. i.e. we set the termination criteria as $\left| f_{optimal} - f_{global} \right| \le VTR$ and record MCN over 30 runs.

Acceleration rate (AR) in %: This criterion is used to compare the convergence speeds between ABC, PSO and ISBC. It is defined as follows:

$$AR = \frac{MCN_{one algorithm} - MCN_{other algorithm}}{MCN_{one algorithm}} \%$$

$$\text{Success Rate (SR)} = \frac{(\#\text{of successof runs})}{total\ runs} \times 100$$

Begin
1. Initialize the population of food sources x_i, using Pseudocode given in Fig 1.
2. Selecting SN fittest food sources from the set of $\{x_{ij} \cup S_{ij}\}$
3. Evaluate each food source x_i, $i = 1,\ldots,SN$
4. *cycle* = 1

Repeat
For each food source $x_i$in the population
5. Generate a new food source V_i by its corresponding employed bee using equation

$$V_{i,j} = w*V_{i,j} + c_1*r_1*(pbest_{i,j} - X_{i,j}) + c_2*r_2*(gbest_{i,j} - X_{i,j})$$
$$X_{ij} = X_{ij} + V_{ij}$$

where w represents the inertia weight, C_1, C_2 are learning factors, r_1, r_2 are independent random numbers uniformly distributed in the range[0,1]. V_i, X_i, $pbest_i$ are the velocity, position and the personal best of the ith particle in the j_{th} dimension respectively. The $gbest_i$ is the jth dimension of best particle in the swarm.

6. Evaluate V_i
7. if $f(V_i) < f(X_i)$ then
8. $X_i = V_i$

End
9. Select, based on fitness proportional selection, the food sources to be visited by onlooker bees

For each food source $x_i$chosen by an onlooker bee
10. Generate a new food source V_i by its corresponding onlooker bee (*Eq. 2*)
11. Evaluate V_i
12. if $f(V_i) < f(X_i)$ then
13. $X_i = V_i$

End
14. Use the scout bee to replace those abandoned food sources

$$X_{ij} = x_{tb} + rand()(x_{r2} - x_{r3})$$

15. Evaluate X_i
16. if $f(V_i) < f(X_i)$ then
17. $X_i = V_i$
18. Save in memory the best food source so far
19. *cycle* = *cycle* + 1
20. **Until** *cycle*
21. **End**

Fig 3: Proposed Algorithm: ISBC

Table 1: Definitions of various chaotic maps

Name	Definition
Logistic Map	$X_{n+1} = 4X_n(1 - X_n)$
Circle Map	$X_{n+1} = X_n + 1.2 - (0.5/2\pi)\sin(2\pi X_n) \bmod(1)$
Gauss Map	$X_{n+1} = \begin{cases} 0, & X_n = 0 \\ 1/X_n \bmod(1), & X_n \in (0,1), \end{cases} \quad 1/X_n \bmod(1) = \frac{1}{X_n} - \left\lfloor \frac{1}{X_n} \right\rfloor$

Henon Map	$X_{n+1} = 1 - 1.4X_n^2 + 0.3X_{n-1}$
Sinus Map	$X_{n+1} = 2.3(X_n)^{2Sin(\pi X_n)}$
Sinusoidal Iterator	$X_{n+1} = Sin(\pi X_n)$
Tent Map	$X_{n+1} = \begin{cases} X_n / 0.7, & X_n < 0.7 \\ 10/3X_n(1 - X_n), & Otherwise \end{cases}$

6. SIMULATION RESULTS

In Table-4 we have recorded the results on the basis of average error. In this case MCN is fixed at 8000,to estimate the average of minimum fitness function value in 30 runs. From the Table-4 it can be clearly observed that for all the 6 shifted benchmark functions ISBC gives better results than ABC and PSO.

In the Table-2, we fixed VTR as given in experimental setting and then calculated the MCN of 30 runs. From Table-2 we can see that the proposed ISBC gives the better results for every function in the comparison to the other algorithms. Further it from Table-2 it is clear that the proposed ISBC is faster than PSO by 40% and ABC by 26%, when D taken as 10 and in other case when D taken as 30, ISBC again performs faster than PSO by 32% and ABC by 16%. The success rates of all the algorithms are given in Table 3, which again demonstrates the efficiency of the proposed algorithm. MCN taken to estimate the average of minimum fitness function value in 30 runs are also presented graphically in Fig. 4.

7. CONCLUSION AND FUTURE WORK

In the present work we have proposed an algorithm, ISBC to improve the convergence speed and to balance the exploration and exploitation. In this paper we made changes in both employed bee phase as well as the scout bee phase. In addition to this the initial population is generated by the combination of uniformly generated random numbers (rand()) & using chaotic systems. The proposed algorithm is tested on a small bed of shifted benchmark functions. Numerical results indicate that the proposed enhancements improve the performance of basic ABC in terms of convergence rate and fitness function. The future work includes to implementation of this algorithm to real life engineering design problems

Table 2: Comparison of the, PSO, ABC and ISBC Algorithms in terms of MCN to VTR and AR(%), (here 3/1 implies ISBC /PSO, and 3/2 implies ISBC /ABC)

Shifted Functions	**F**	**PSO**	**ABC**	**ISBC**	**PSO**	**ABC**	**ISBC**	**3/1**		**3/2**	
		D = 10			*D = 30*			*D = 10*		*D = 30*	
Sphere	F_1	430	379	260	7807	5477	4329	0.40	0.31	0.45	0.21
Schwefel's	F_2	432	314	196	19901	17493	14968	0.55	0.38	0.25	0.14
Rosenbrock's	F_3	9810	7639	6814	17495	16019	13714	0.31	0.11	0.22	0.14
Rastrigin's	F_4	7915	7241	6583	21084	19930	15932	0.17	0.09	0.24	0.20
Griekwank's	F_5	852	690	469	2161	1545	1367	0.45	0.32	0.37	0.12
Ackley's	F_6	859	567	378	1891	1393	1185	0.56	0.33	0.37	0.15

Table 3: Success rate

F	**PSO**	**ABC**	**ISBC**	**PSO**	**ABC**	**ISBC**
	D = 10			*D = 30*		
F_1	100	100	100	100	100	100
F_2	20	30	30	20	20	30
F_3	20	20	30	10	20	30
F_4	30	20	40	30	30	30
F_5	40	40	60	50	40	50
F_6	100	100	100	100	100	100

Table 4: Comparison of proposed ISBC algorithm with PSO, ABC on shifted functions in terms of error (best and mean) and standard deviation (SD) for 10 and 30 dimensions.

F	Error Value	PSO	ABC	ISBC	PSO	ABC	ISBC
		Dimension = 10			*Dimension = 30*		
F_1	best	3.709E-07	2.901E-07	1.674E-07	6.811E-07	6.145E-07	6.090E-07
	mean	7.518E-07	7.393E-07	5.279E-07	8.878E-07	8.536E-07	8.192E-07
	SD (±)	1.785E-07	2.091E-07	1.030E-07	1.122E-07	1.057E-07	1.022E-07
	MCN	4.300E+02	3.790E+02	2.600E+02	7.807E+03	5.477E+03	4.329E+03
F_2	best	9.810E-07	9.129E-07	8.093E-07	5.164E+00	2.308E-03	2.092E-03
	mean	4.063E-01	8.391E-07	6.926E-07	1.639E+01	3.389E+00	2.912E+00
	SD(±)	5.824E-01	8.184E-07	7.939E-07	7.070E+00	4.285E+00	2.093E+00
	MCN	4.320E+02	3.140E+02	1.960E+02	8.000E+03	8.000E+03	8.000E+03
F_3	best	2.931E+00	2.193E+00	1.294E+00	1.571E+02	1.427E+02	1.379E+02
	mean	5.532E+00	4.954E+00	4.001E+00	1.959E+02	1.860E+02	1.549E+02
	SD(±)	1.519E+00	2.937E+00	3.376E+00	1.738E+02	1.688E+02	1.690E+02
	MCN	8.000E+03	7.639E+03	6.814E+03	8.000E+03	8.000E+03	8.000E+03
F_4	best	3.583E-07	3.138E-07	1.039E-07	2.985E+00	1.990E+00	1.009E+00
	mean	3.064E-01	1.288E-01	2.000E-05	3.683E+00	2.892E+00	2.548E+00
	SD(±)	4.513E-01	1.838E-01	8.746E-02	3.002E+00	1.900E+00	1.213E+00
	MCN	7.915E+03	7.241E+03	6.583E+03	8.000E+03	8.000E+03	8.000E+03
F_5	best	4.550E-07	3.183E-07	1.274E-07	7.528E-07	6.783E-07	6.193E-07
	mean	3.258E-03	5.028E-07	4.290E-07	8.946E-07	8.746E-07	8.314E-07
	SD(±)	3.931E-03	5.198E-07	3.787E-07	7.393E-08	9.781E-08	8.280E-07
	MCN	8.520E+02	6.900E+02	4.690E+02	2.161E+03	1.545E+03	1.367E+03
F_6	best	4.017E-07	3.376E-07	1.834E-07	8.540E-07	7.716E-07	7.450E-07
	mean	6.894E-07	6.081E-07	3.875E-07	9.171E-07	9.122E-07	9.292E-07
	SD(±)	1.399E-07	3.930E-07	2.742E-07	3.198E-08	6.963E-08	5.575E-07
	MCN	8.590E+02	5.670E+02	3.780E+02	1.891E+03	1.393E+03	1.185E+03

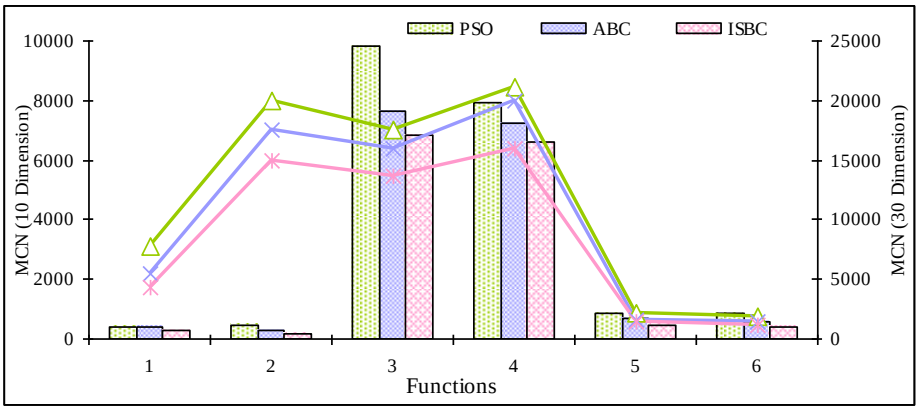

Fig 4: MCN taken by 6 Shifted benchmark functions with D=10 and 30 by PSO, ABC and ISBC

8. REFERENCES

[1] D. Karaboga, An Idea based on Bee Swarm for Numerical Optimization, Tech. Rep. TR-06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[2] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.

[3] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (1) (2008) 687–697.

[4] Q.K. Pan, M.F. Tasgetiren, P.N. Suganthan, T.J. Chua, A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Information Sciences 181 (12) (2011) 2455–2468.

[5] S. Sundar, A. Singh, A swarm intelligence approach to the quadratic minimum spanning tree problem, Information Sciences 180 (17) (2010) 3182–3191.

[6] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers & Structures 87 (13-14) (2009) 861–870.

[7] Karaboga D, Basturk B. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation 2009;214:108–32.

[8] Zhu GP, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Computation 2010, doi:10.1016/j.amc.2010.08.049.

[9] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers & Structures 87 (13-14) (2009) 861–870.

[10] Eberhart, R., Kennedy, J., A New Optimizer using Particle Swarm Theory. In: Proceedings of 6th International Symposium on Micro Machine and Human Science (MHS), Cape Cod, MA, November, pp. 39–43 (1995).

[11] Eberhart, R., Kenedy, J., Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, November, pp.1114–1121, (1995).

[12] Alatas, B., Akin, E., & Ozer, B. (2009). Chaos embedded particle swarm optimization algorithms. Chaos, Solitons & Fractals. doi:10.1016/j.chaos.2007.09.063.

[13] Coelho, L. S., & Mariani, V. C. (2008). Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. Expert Systems with Applications, 34, 1905–1913.

[14] Heidari-Bateni, G., & McGillem, C. D. (1994). A Chaotic direct-sequence spread spectrum communication system. IEEE Transaction on Communications, 42(2/3/4), 1524–1527.

[15] Suneel, M. (2006). Chaotic sequences for secure CDMA. Ramanujan Institute for Advanced Study in Mathematics, 1–4.

[16] Wong, K., Man, K. P., Li, S., & Liao, X. (2005). More secure chaotic cryptographic scheme based on dynamic look-up table circuits. Systems & Signal Processing, 24(5), 571–584.

[17] Arena, P., Caponetto, R., Fortuna, L., Rizzo, A., & La Rosa, M. (2000). Self organization in non recurrent complex system. International Journal of Bifurcation and Chaos, 10(5), 1115–1125.

[18] Manganaro, G., & de Gyvez, J. P. (1997). DNA computing based on chaos. In IEEE International conference on evolutionary computation (pp. 255–260). Piscataway, NJ: IEEE Press.

[19] Han, F., Hu, J., Yu, X., & Wang, Y. (2007). Fingerprint images encryption via multiscroll chaotic attractors. Applied Mathematics and Computation, 185(2), 931–939.

[20] Alatas B. Chaotic bee colony algorithms for global numerical optimization. Expert Systems with Applications 2010;37: 5682–7.

[21] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, 2005.