JAISCR, 2012, Vol. 2, No. 2, pp. 147 – 155

ON THE HYBRIDIZATION OF THE ARTIFICIAL BEE COLONY AND PARTICLE SWARM OPTIMIZATION ALGORITHMS

Mohammed El-Abd
Computer Engineering, American University of Kuwait, Kuwait

Abstract

In this paper we investigate the hybridization of two swarm intelligence algorithms; namely, the Artificial Bee Colony Algorithm (ABC) and Particle Swarm Optimization (PSO). The hybridization technique is a component-based one, where the PSO algorithm is augmented with an ABC component to improve the personal bests of the particles. Three different versions of the hybrid algorithm are tested in this work by experimenting with different selection mechanisms for the ABC component. All the algorithms are applied to the well-known CEC05 benchmark functions and compared based on three different metrics, namely, the solution reached, the success rate, and the performance rate.

1 Introduction

Both the artificial bee colony (ABC) and particle swarm optimization (PSO) algorithms are two population-based algorithms developed in the past 15 years. Both algorithms are nature-inspired as PSO mimics the behavior of a group of birds or a school of fish looking for food while ABC mimics the behavior of honey bees when locating food sources. ABC and PSO have been proven through many different studies [1, 2, 3, 4, 5] to be very efficient in function optimization and were applied to many engineering applications.

The aim of this work is to combine these two algorithms in order to gain benefit from their good characteristics. In [6], it was shown that ABC has an excellent performance on separable functions and good competitive performance on multi-modal and hybrid functions. On the other hand, it was shown that the standard particle swarm optimization (SPSO) algorithm has the best performance on uni-modal functions.

To the best of our knowledge, the only previous attempt to combine these two algorithms was proposed in [7]. However, the algorithm was a cooperative rather than a hybrid one. The idea was to have two separate ABC and PSO swarms running in parallel and exchanging information during the search. In this work, the hybridization is attempted at the component level in order to come up with a hybrid algorithm.

The paper is organized as follows: Sections 2 and 3 cover the basic ABC and PSO algorithms. The hybrid algorithm is introduced in Section 4. Results are presented and discussed in Section 5.

2 Artificial Bee Colony

The ABC algorithm was first proposed in [8]. The algorithm was inspired by the method adopted by a swarm of honey bees to locate food sources. There are two different honey bee groups that share knowledge in order to successfully locate such sources. First, there are the *employed bees* that are currently exploiting a food source. Second, there are the *unemployed bees* that are continuously looking for a food source. Unemployed bees are divided into *scout bees* that search around the nest and *on-*

lookers that wait at the nest and establish communication with the employed bees.

Algorithm 1 The ABC algorithm

```
Require: $Max\_Cycles, S_n, limit$
 1: Initialize the food sources
 2: Evaluate the food sources
 3: $Cycle = 1$
 4: while $Cycle \leq Max\_Cycles$ do
 5:     Produce new solutions using employed bees as in eq. 1
 6:     Evaluate the new solutions and apply greedy selection
 7:     Calculate the probability using eq. 2
 8:     Produce new solutions using onlooker bees as in eq. 1
 9:     Evaluate the new solutions and apply greedy selection
10:     for All solutions do
11:         if A solution has not been improved for $limit$ cycles then
12:             Generate a new random solution a scout bee as in eq. 3
13:         end if
14:     end for
15:     Memorize the best solution found so far
16:     $Cycle = Cycle + 1$
17: end while
18: return best solution
```

This algorithm was applied to multidimensional and multi-modal function optimization in [8, 9]. The swarm is divided into employed bees, scouts and onlookers. S_n solutions to the problem are randomly initialized in the function domain and referred to as food sources. A number of employed bees, set as the number of food sources and half the colony size, are used to find new food sources using the following equation:

$$v_{ij} = x_{ij} + \phi_{ij} \times (x_{ij} - x_{kj}), \tag{1}$$

where $\mathbf{x}_{ij}$ refers to problem variable j in food source number i. j is a randomly selected number in [1,D] and D is the number of dimensions. ϕ_{ij} is a random number uniformly distributed in the range [-1,1] while k is the index of a randomly chosen solution. Both $\mathbf{v}_i$ and $\mathbf{x}_i$ are then compared against each other and the employed bee exploits the better food source, which is a *greedy selection* mechanism.

Onlooker bees next choose a random food source according to the probability given in equation 2. Then, each onlooker bee tries to find a better food source around the selected one using equation 1.

$$p_i = \frac{fit_i}{\sum_{j=1}^{S_n} fit_j}, \tag{2}$$

where fit_i is the fitness of the i^{th} food source.

Finally, if a certain food source i cannot be improved for a predetermined number of cycles, referred to as $limit$, this food source is abandoned. The employed bee that was exploiting this food source becomes a scout that looks for a new food source by randomly searching the problem domain using the following equation:

$$x_{ij} = lb_j + r \times (ub_j - lb_j), \tag{3}$$

where lb_j and ub_j are the lower and upper bounds for problem variable j, and r is a random number uniformly distributed in the range [0,1]. The ABC algorithm is shown in Algorithm 2.

3 Particle Swarm Optimization

PSO is a population-based method, where the population is referred to as a *swarm*. The swarm consists of a number of individuals called *particles*. Each particle i in the swarm holds: (i) the current position $\mathbf{x}_i$, which represents a solution to the problem, (ii) the current velocity $\mathbf{v}_i$, (iii) the best position $\mathbf{pbest}_i$, the one associated with the best objective function value the particle has achieved so far, and (iv) the neighborhood best position $\mathbf{nbest}_i$, the one associated with the best objective function value found in the particle's neighborhood. The choice of $\mathbf{nbest}_i$ depends on the neighborhood topology adopted by the swarm, different neighborhood topologies have been studied in [10].

In traditional PSO, each particle adjusts its own position in every iteration in order to move towards its best position and the neighborhood best according to the following equations:

$$\begin{aligned} v_{ij}^{t+1} = wv_{ij}^{t} + c_1 r_1 (pbest_{ij}^{t} - x_{ij}^{t}) \\ + c_2 r_2 (nbest_{ij}^{t} - x_{ij}^{t}), \end{aligned} \tag{4}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1}, \tag{5}$$

for $j \in \{1 \ldots D\}$ where D is the number of dimensions, $i \in \{1 \ldots n\}$ where n is the number of particles, t is the iteration number, w is the inertia weight, r_1 and r_2 are two random numbers uniformly distributed in the range [0,1), and c_1 and c_2 are the acceleration factors.

After changing its position, each particle updates its personal best position using (assuming a minimization problem):

$$\mathbf{pbest}_i^{t+1} = \begin{cases} \mathbf{pbest}_i^t & \texttt{if} \quad f(\mathbf{pbest}_i^t) \leq f(\mathbf{x}_i^{t+1}), \\ \mathbf{x}_i^{t+1} & \texttt{if} \quad f(\mathbf{pbest}_i^t) > f(\mathbf{x}_i^{t+1}). \end{cases} \tag{6}$$

Finally, the global best of the swarm is updated using the following equation:

$$\mathbf{gbest}^{t+1} = \arg \min_{\mathbf{pbest}_i^{t+1}} f(\mathbf{pbest}_i^{t+1}). \tag{7}$$

This model is referred to as the *lbest* (local best) model. Another simple model is the *gbest* (global best) model, which is the case when the particle's neighborhood is defined as the whole swarm. The basic PSO algorithm is shown in Algorithm 3.

Algorithm 2 The PSO algorithm

```
Require: $Max\_Function\_Evaluations, n, w, c_1, c_2$
 1: Initialize the swarm
 2: Evaluate the swarm
 3: $Max\_Iterations = \frac{Max\_Function\_Evaluations}{Num\_Particles}$
 4: $Iter\_number = 1$
 5: while $Iter\_number \leq Max\_Iterations$ do
 6:     for every particle $i$ do
 7:         Update $\mathbf{v}_i$ as in eq. 4
 8:         Update $\mathbf{x}_i$ as in eq. 5
 9:         Update $\mathbf{pbest}_i$ as in eq. 6
10:     end for
11:     Update gbest
12:     $Iter\_number = Iter\_number + 1$
13: end while
14: return gbest as in eq. 7
```

4 The Hybrid Algorithm

The aim behind the hybrid algorithm is to effectively mix components from both ABC and SPSO in order to have an algorithm that easily solves separable problems as ABC while having a rotationally invariant behavior as SPSO at the same time. The reason for ABC being so good on separable functions is that its update equation only updates a single problem variable at a time after which the new solution is re-evaluated.

4.1 Mixing Approach

The ABC component is added to SPSO after the main loop. For m trials, a particle i is selected, and a new candidate solution $\mathbf{v}$ is produced using the ABC update equation. This is done after randomly selecting another particle k as a neighbor and a random problem variable j, hence $\mathbf{v}$ is produced as follows:

$$\mathbf{v}_l = \begin{cases} \mathbf{pbest}_{il}, & l \neq j, \\ \mathbf{pbest}_{il} + \phi_{il} \times (\mathbf{pbest}_{il} - \mathbf{pbest}_{kl}), & l = j. \end{cases} \tag{8}$$

The new candidate solution $\mathbf{v}$ replaces $\mathbf{pbest}_i$ if it has a better fitness. The steps for this hybrid approach is shown in Algorithm 4.1.

Algorithm 3 The ABC-PSO algorithm

```
Require: $Max\_Function\_Evaluations, n, w, c_1, c_2, m$
 1: Initialize the swarm
 2: Evaluate the swarm
 3: Max_Iterations = $\frac{Max\_Function\_Evaluations}{Num\_Particles}$
 4: Iter_number=1
 5: while $Iter\_number \leq Max\_Iterations$ do
 6:     for every particle $i$ do
 7:         Update $\mathbf{v}_i$
 8:         Update $\mathbf{x}_i$
 9:         if $f(\mathbf{pbest}_i) \leq f(\mathbf{x}_i)$ then
10:             $\mathbf{pbest}_i = \mathbf{x}_i$
11:         end if
12:     end for
13:     Update gbest
14:     for $m$ trials do
15:         Select a particle $i$ to improve
16:         Select a different random particle $k$
17:         Select a random problem variable $j$
18:         Apply ABC update rule to $\mathbf{pbest}_i$ as in eq. 8
19:         Update $\mathbf{pbest_i}$ and gbest
20:     end for
21:     $Iter\_number = Iter\_number + 1$
22: end while
23: return gbest
```

4.2 Effect of m and the selection scheme

The number of trials m controls how much computational power is given to the ABC component in the hybrid algorithm. For example, if m equals the number of particles in the swarm, both SPSO and the ABC component have an equal share of the allowed number of function evaluations.

Another design decision that affects the performance of the hybrid algorithm is how to select a particle to update. Two appropriate selection schemes are:

– Random selection, will be referred to as **Hybrid** in the rest of the paper,
– Fitness proportionate selection:
  – Higher probability of selecting good particles, **Hybrid**$_{FP}$,
  – Higher probability of selecting bad particles, **Hybrid**$_{IFP}$.

For fitness proportionate selection, more update attempts could be either targeted towards good or bad particles. As for random selection, it becomes only necessary if $m < n$ (the swarm size). However, if $m = n$, an update attempt is carried out for every particle.

5 Results and Discussion

5.1 Experimental Setup

All the algorithms are applied to the CEC05 benchmark functions [11]. This library provides a class of shifted and/or rotated functions including uni-modal functions (f1-f5) and multi-modal functions (f6-f14). For all experiments, the termination criterion is the maximum number of allowable function evaluations set as $10^4 \times D$, where D is the problem dimensionality. Experiments are carried out for D = 10, 30 and 50 dimensions. For all algorithms, initial candidate solutions are randomly initialized using uniform distribution over the specified domain. For each function, the reported solution is the average taken over all 30 different runs.

Performance assessment is based on three metrics. First, the solution reached after the allocated number of function evaluations. Second, the success rate defined as the number of successful runs over the total number of runs (a successful run is defined as a run where the tested algorithm has reached a predefined threshold for the function under study). Third, the performance rate, which is defined as follows:

$$Performance\ rate = \frac{FEV\_avg \times total\_runs}{successful\_runs}, \quad (9)$$

where FEV_avg is the average number of function evaluations needed to reach the predetermined threshold taken over the successful runs only. The threshold values are defined in [11] as 10^{-6} for uni-modal functions and 10^{-2} for multi-modal functions.

Experiments use the ABC code available at [12] and SPSO code (version 2007) available at [13]. The hybrid algorithm is implemented by augmenting the available SPSO code with the ABC component.

5.2 Parameter Tuning

For ABC, the work in [14] indicated that there is no need to have a huge colony size in order to provide good results. The experiments were repeated using populations of 20, 40 and 100 bees for the three problem sizes. It was found that using a swarm of 40 bees provided the best results on average for all the dimensions. The recommendations in [15] were followed by setting the *limit* parameters to $S_n \times D$, although recent research [14] indicated that lower values might be needed for more difficult functions.

For SPSO, the only parameter set is the swarm size and it was set to 40. The parameter values for w, c_1, and c_2 are already set in the code as 1.193, 1.193, and 0.721 respectively.

5.3 Experimental Results

The results are provided in Table 1 for uni-modal functions, and Tables 2 and 3 for multi-modal functions. The best results are highlighted in bold (to test for the significance of the results, we used the Mann-Whitney non-parametric statistical test, where the null hypothesis is rejected with a 95% confidence level).

For uni-modal functions, the results in Table 1 show that all versions of the hybrid algorithm have a similar performance for a problem size of 10. For higher problem sizes, there is no clear version that comes out as the best performer. While the **Hybrid** version has a better performance on f5, **Hybrid**$_{FP}$ has the better performance on f4 and **Hybrid**$_{IFP}$ has the better performance on f2.

For multi-modal functions, the results in Tables 2 and 3 show that the Hybrid version is the best performer for a problem size of 10. The same behavior continues for higher problem sizes with the excep-

Table 1. Results of all the algorithms for the uni-modal CEC05 functions

Benchmark Function	**Size**	**ABC**	**SPSO**	**Hybrid**	**Hybrid**$_{FP}$	**Hybrid**$_{IFP}$
f1	10	**0**	**0**	***0***	***0***	***0***
		0	**0**	***0***	***0***	***0***
f2		3.73	**0**	***0***	***0***	***0***
		2.52	**0**	***0***	***0***	***0***
f3		6.41e+05	**5.64e+04**	9.85e+04	1.02e+05	9.42e+04
		2.78e+05	**3.27e+04**	6.47e+04	8.65e+04	7.57e+04
f4		29.17	**0**	***0***	***0***	***0***
		41.83	**0**	***0***	***0***	***0***
f5		86.76	**0**	***0***	***0***	***0***
		100.53	**0**	***0***	***0***	***0***
f1	30	1.06e-13	**0**	***0***	***0***	***0***
		1.97e-14	**0**	***0***	***0***	***0***
f2		2.49e+03	**0**	***0***	***0***	***0***
		969.89	**0**	***0***	***0***	***0***
f3		6.22e+06	**2.21e+05**	8.02e+05	6.83e+05	*4.68e+05*
		1.67e+06	**9.53e+04**	1.07e+06	4.30e+05	*1.58e+05*
f4		1.49e+04	**8.35e-04**	8.88	*2.26*	3.27
		5.15e+03	**1.29e-03**	5.31	*2.03*	3.49
f5		1.08e+04	3.44e+03	***3.15e+03***	3.42e+03	3.68e+03
		1.47e+03	777.42	***751.01***	668.98	941.01
f1	50	2.04e-13	**0**	***0***	***0***	***0***
		3.20e-14	**0**	***0***	***0***	***0***
f2		1.61e+04	**0**	8.00e-07	1.17e-06	***0***
		4.05e+03	**0**	6.10e-07	1.12e-06	***0***
f3		1.08e+07	**2.59e+05**	*4.79e+05*	8.68e+05	6.00e+05
		2.65e+06	**7.85e+04**	*1.35e+05*	3.94e+05	2.98e+05
f4		6.42e+04	**152.51**	3.05e+03	*1.52e+03*	*1.56e+03*
		1.24e+04	**118.64**	1.23e+03	*636.58*	*783.63*
f5		2.54e+04	8.84e+03	***8.16e+03***	***8.30e+03***	9.06e+03
		2.54e+03	1.59e+03	***1.16e+03***	***1.25e+03***	1.50e+03

Table 2. Results of all the algorithms for the multi-modal CEC05 functions, 10 and 30 dimensions

Benchmark Function	**Size**	**ABC**	**SPSO**	**Hybrid**	**Hybrid$_{FP}$**	**Hybrid$_{IFP}$**
f6	10	**1.46**	39.89	*14.90*	437.75	271.77
		1.63	113.23	*56.89*	1.12e+03	663.83
f7		2.45e-01	4.36e-02	***2.75e-02***	6.55e-02	5.08e-02
		1.27e-01	1.96e-02	***1.43e-02***	4.55e-02	3.42e-02
f9		**0**	5.26	***0***	***0***	***0***
		0	2.74	***0***	***0***	***0***
f10		28.59	**4.59**	6.11	6.47	6.34
		8.29	**1.96**	2.11	2.29	2.97
f11		5.38	**2.97**	4.67	4.11	4.89
		6.04e-01	**1.52**	7.85e-01	2.11	1.61
f12		**299.39**	5.97e+03	***210.31***	1.40e+03	641.68
		177.41	3.00e+03	***117.38***	1.05e+03	722.03
f13		**2.19e-01**	8.60e-01	*2.76e-01*	4.23e-01	4.18e-01
		8.95e-02	2.18e-01	*5.72e-02*	1.01e-01	1.92e-01
f14		3.34	**2.47**	2.76	2.85	2.82
		2.12e-01	**3.93e-01**	3.76e-01	3.55e-01	3.72e-01
f6	30	**7.49**	141.65	*341.18*	549.57	1.93e+03
		10.99	226.24	*364.37*	1.32e+03	3.15e+03
f7		**1.23e-02**	2.04e-02	2.61e-02	*2.24e-02*	3.62e-02
		5.04e-03	1.57e-02	2.67e-02	*2.71e-02*	4.98e-02
f9		6.06e-14	44.37	***0***	***0***	6.23
		1.44e-14	10.15	***0***	***0***	6.25
f10		333.43	**46.13**	67.80	*59.08*	68.78
		68.97	**11.91**	14.42	*22.62*	28.93
f11		**27.94**	30.06	***28.09***	31.31	30.11
		1.62	2.85	***2.26***	2.21	3.39
f12		8.55e+03	2.95e+05	1.45e+04	2.20e+04	***7.98e+03***
		3.70e+03	6.28e+04	4.27e+03	1.03e+04	***4.75e+03***
f13		**9.44e-01**	4.37	*1.30*	1.51	1.78
		1.19e-01	1.27	*1.46e-01*	2.25e-01	6.12e-01
f14		12.97	**12.44**	***12.33***	12.63	12.66
		2.16e-01	**3.76e-01**	***3.53e-01***	3.61e-01	3.53e-01

Table 3. Results of all the algorithms for the multi-modal CEC05 functions, 50 dimensions

Benchmark Function	**Size**	**ABC**	**SPSO**	**Hybrid**	**Hybrid$_{FP}$**	**Hybrid$_{IFP}$**
f6	50	**6.16**	219.74	*272.91*	990.98	886.54
		7.76	243.62	*324.18*	2.16e+03	1.84e+03
f7		**1.68e-04**	9.66e-03	1.24e-02	1.07e-02	*3.85e-03*
		1.23e-04	1.61e-02	1.62e-02	1.55e-02	*8.65e-03*
f9		**1.14e-13**	119.56	*1.22e-03*	1.33e-01	50.88
		1.02e-28	25.62	*6.69e-03*	3.44e-01	31.24
f10		1.08e+03	**130.47**	182.04	***135.08***	209.53
		102.68	**25.06**	32.32	***53.94***	66.77
f11		**56.24**	**57.28**	***57.15***	60.54	60.16
		2.23	**3.84**	***2.07***	4.07	3.59
f12		**3.52e+04**	1.60e+06	6.80e+04	6.71e+04	***3.39e+04***
		1.05e+04	2.94e+05	1.66e+04	2.23e+04	***1.83e+04***
f13		**1.63**	9.02	*2.69*	*2.67*	5.36
		1.97e-01	3.51	*2.06e-01*	*3.72e-01*	2.13
f14		22.68	**22.01**	***21.87***	22.26	22.22
		2.05e-01	**4.33e-01**	***3.12e-01***	3.87e-01	3.64e-01

Table 4. Performance rates and success rates of all algorithms for the CEC05 functions successfully solved

	Size	No. Solved Functions	Normalized Performance Rates						
			F1	**F2**	**F4**	**F5**	**F6**	**F7**	**F9**
Best Performance Rates			5852	12837	15400	11880	-	544500	4757
ABC		2	1.75 100%	-	-	-	-	-	2.38 100%
SPSO		4	1 100%	1 100%	1 100%	1 100%	-	-	-
Hybrid	10	6	1.89 100%	1.82 100%	1.85 100%	1.78 100%	-	1 13.33%	7.24 100%
Hybrid$_{FP}$		6	1.13 100%	1.69 100%	1.87 100%	1.68 100%	-	3.31 3.33%	2.02 100%
Hybrid$_{IFP}$		6	1.04 100%	1.51 100%	1.85 100%	1.78 100%	-	1.35 6.67%	1 100%
Best Performance Rates			18873	97200	-	-	4718313	58320	41594
ABC		4	3.58 100%	-	-	-	1 3.33%	21.23 20%	1.39 100%
SPSO		3	1.49 100%	1 100%	-	-	-	1 50%	-
Hybrid	30	4	2.67 100%	1.86 100%	-	-	-	2.27 40%	5.01 100%
Hybrid$_{FP}$		4	1.72 100%	1.81 100%	-	-	-	1.4 53.33%	1 100%
Hybrid$_{IFP}$		4	1 100%	1.4 100%	-	-	-	1.26 30%	17.17 13.33%
Best Performance Rates			28222	280160	-	-	-	52186	107912
ABC		3	3.47 100%	-	-	-	-	6.06 100%	1 100%
SPSO		3	1.39 100%	1 100%	-	-	-	1.05 100%	-
Hybrid	50	4	2.53 100%	5.83 30%	-	-	-	2.67 66.67%	4.02 96.67%
Hybrid$_{FP}$		4	1.67 100%	5.75 30%	-	-	-	1.77 66.67%	1.07 86.67%
Hybrid$_{IFP}$		3	1 100%	1.28 100%	-	-	-	1 86.67%	-

tion of **Hybrid**$_{FP}$ being the best performer on f10 and **Hybrid**$_{IFP}$ being the best performer on f12.

Inspecting the results in Table 4 shows that all versions of the hybrid algorithm are able to reach the predetermined threshold for more functions than ABC or SPSO. For all the successfully solved functions, the **Hybrid** version almost has double the speed of convergence of SPSO (measured in terms of consumed function evaluations). Among the hybrid algorithms, the **Hybrid**$_{IFP}$ version has the fastest speed of convergence.

The results of all the algorithms for f1 and f9 show that the **Hybrid** version was the one able to maintain the good performance of ABC on separable functions both from the solution reached point of view as well as the success rate.

6 Conclusion

In this work we proposed a hybrid algorithm consisting of ABC and SPSO components. This is achieved by augmenting the SPSO algorithm with an ABC component. This ABC component updates the **pbest** information of the particles in every iteration using the ABC update equation. Three different selection mechanisms are tested within the ABC component. One approach updates all the particles using the ABC component while the remaining two update the particles in a fitness proportionate manner.

The algorithms were tested using the CEC05 benchmark library and it was shown that updating all particles is usually able to provide better solutions than the versions using the fitness proportionate approach.

The hybrid algorithms were able to successfully solve more problems than either ABC or SPSO alone, although they may suffer from a lower success rate in some situations. Another observation is that the hybrid algorithms have a slower speed of convergence than SPSO. Applying the update component to all particles had the worst effect on the speed of convergence. This is followed by selecting the particles in a fitness proportionate approach and finally by the inverse fitness proportionate selection scheme.

In comparison to ABC, the most successful hybrid algorithm was the one involving updating all the particles as it had the highest success rate of all the hybrid versions on separable functions. Again, this comes with the expense of having a slower speed of convergence in comparison with ABC.

In future work it is intended to test the effect of changing the parameter m on the performance of the hybrid algorithm. We also intend to test how to incorporate more than a single selection scheme within the ABC component. A third direction is to further augment the SPSO algorithm with the ABC re-initialization component for a better chance of escaping local minima.

References

[1] D. Karaboga and B. Akay, “A survey: Algorithms simulating bee swarm intelligence,” *Artificial Intelligence Review*, vol. 31, pp. 61–85, 2009.

[2] S. Bitam, M. Batouche, and E. Talbi, “A survey on bee colony algorithms,” in *Proceedings of 24th IEEE/ACM International Parallel and Distributed Processing Symposium IPDPS*, 2010, pp. 1–8.

[3] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization: An overview,” *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[4] A. Banks, J. Vincent, and C. Anyakoha, “A review of particle swarm optimization. part i: Background and development,” *Natural Computing*, vol. 6, pp. 467–484, 2007.

[5] ——, “A review of particle swarm optimization. part ii: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications,” *Natural Computing*, vol. 7, pp. 109–124, 2008.

[6] M. El-Abd, “Performance assessment of foraging algorithms vs. evolutionary algorithms,” *Information Sciences*, doi:10.1016/j.ins.2011./09.005.

[7] X. shi, Y. Li, H. Li, R. Guan, L. Wang, and Y. Liang, “An integrated algorithm based on artificial bee colony and particle swarm optimization,” in *Proc. of 6th International Conference on Neural Computation*, 2010, pp. 2586–2590.

[8] D. Karaboga, “An idea based on honey bee swarm for numerical optimization,” Engineering Faculty, Computer Engineering Department, Erciyes University, Tech. Rep. TR06, 2005.

[9] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm.” *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[10] J. Kennedy and R. Mendes, “Population structure and particle swarm performance,” in *Proc. of IEEE Congress on Evolutionary Computation*, vol. 2. Washington, D.C.:IEEE Computer Society, 2002, pp. 1671–1676.

[11] 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 CEC 2005 special session on real-parameter optimization,” ITT Kanpur, India, Tech. Rep. 2005005, 2005.

[12] D. Karaboga, “Artificial bee colony code,” http://mf.erciyes.edu.tr/abc/software.htm, 2008.

[13] Particle Swarm Central, “Standard pso 2007 code,” http://www.particleswarm.info, 2007.

[14] B. Akay and D. Karaboga, “Parameter tuning for the artificial bee colony algorithm,” in *Proceedings of 1st International Conference on Computational Collective Intelligence*. Springer-Verlag:Berlin Heidelberg, 2009, pp. 608–619.

[15] D. Karaboga and B. Akay, “A comparative study of artificial bee colony algorithm,” *Applied Mathematics and Computation*, vol. 214, pp. 108–132, 2009.