A Powerful Bee Swarm Optimization Algorithm

Reza Akbari, Alireza Mohammadi, and Koorush Ziarati
Department of Computer Science and Engineering, Shiraz University, Shiraz, Iran
{rakbari, a_mohammadi}@cse.shirazu.ac.ir, ziarati@shirazu.ac.ir

***Abstract*-The optimization algorithms which are inspired from intelligent behavior of honey bees are among the most recently introduced techniques. In this paper, a novel algorithm called bee swarm optimization, or BSO, is presented. The BSO is a population based optimization technique which is inspired from foraging behavior of honey bees. The proposed approach provides different patterns which are used by the bees to adjust their flying trajectories. The BSO algorithm is compared with existing bee algorithms on a set of well known numerical test functions. The experimental results show that the BSO algorithm is effective and robust; produce excellent results, and outperform other algorithms investigated in this consideration.**

I.. INTRODUCTION

Optimization has been an active area of research for several decades. As many real-world optimization problems become increasingly complex, better optimization algorithms are always needed. In optimization problems, the objective is to find the minimum or maximum of the function under consideration. There are many population based optimization techniques available for unconstrained numerical optimization. Genetic algorithms (GA), Particle Swarm Optimization (PSO), and Bee Algorithms (BA) are among the most popular optimization algorithms which employ a population of individuals to solve the problem on the hand.

The algorithms which are inspired from intelligent behaviors of honey bees have been developed and applied to different engineering fields [1]-[7]. However, a few algorithms based on this idea were presented in literature for numerical optimization. A numerical optimization algorithm based on foraging behavior of honey bees, called Artificial Bee Colony (or ABC) was proposed by Karaboga in [8]. In ABC, the employer bees try to find food source and advertise them. The onlooker bees follow their interesting employer, and the scout bee fly spontaneously to find better food sources. A virtual bee algorithm (or VBA) developed by Yang in [9]. The VBA aimed to optimize the numerical function in 2 dimensions using a swarm of virtual bees which move randomly in the phase space and interact by finding food sources corresponding to the encoded values of the function. The intensity of interactions between these bees results the solution for the optimization problem. Sundareswaran et al. proposed a different approach based on natural behavior of honey bees in nectar collection in which the randomly generated worker bees are forced to move in the direction of the elite bee. The elite bee represent the best possible solution [10]. The bees move based on a probabilistic approach. The step distance of flight of bees is made as a variable parameter in the algorithm. The experiments showed that developed algorithms based on intelligent behavior of honey bees are successful in solving numerical optimization problems and outperform other population based algorithm such as PSO, GA, and ACO [8]-[11].

This work presents a novel method based on foraging behaviors of honey bee swarms. The proposed approach uses different types of bees to optimize numerical functions. Each type of bees employs a distinct moving pattern. The scout bees fly randomly over their nearby regions. An onlooker bee selects an experienced forager bee as its interesting elite and moves toward that bee. An experienced forager bee memorize the information about the best food source found so far by that bee, selects the best experienced forager bee as the elite bee, and adjust its position based on these information.

The paper is organized as follows. Section 2 introduces the intelligent behaviors of honey bees. Description of the proposed BSO algorithm is presented in section 3. Section 4 reports experimental analysis on the proposed approach in comparison with other algorithms. Finally, section 5 concludes this work.

II. INTELLIGENT BEHAVIORS OF HONEY BEES

A colony of honey bees capable to perform complex tasks using relatively simple rules of individual bees' behavior. Collecting, processing, and advertising of nectars are examples of intelligent behaviors of honey bees [5]. A bee may have one of the following types: (employed / unemployed) forager, scout, onlooker, recruit, and experienced forager [3]-[6]. The type of a bee depends on action which is performed by the bee and the level of information which may be used by the bee. A potential forager will initialize as unemployed forager. This type of bees has no knowledge about the environment, and the position of food sources in the environment. There are two types of unemployed foragers with different flying patterns, so-called scout and onlooker bees. A scout bee fly spontaneously around the hive and search for new food sources without any knowledge about the environment. The

978-1-4244-4873-9/09/$25.00 ©2009 IEEE

onlooker bees wait in the nest, process the information shared by employed foragers, and select their interesting dancers. After that, the unemployed forager can be a recruit and can start searching by using the knowledge from the selected dancer bee.

The employed forager bees provide information about the environment and the currently discovered food sources for the onlooker bees and advertise them on the dance floor. The provided information by employed foragers is shared with a probability proportional to the profitability of the food source. So, the onlooker bees can employ a probabilistic approach to choose an employed forager between numerous dancers and adjust its search trajectory toward the most profitable source. The onlooker bees choose more profitable food sources with a greater probability, since more profitable sources provide more valuable information. Hence, more profitable food sources attract more recruit bees.

After choosing the food source by the recruit bee, she flies to finds the food source. After finding the food source, the recruit bee switches its type to employed forager. The employed forager bee memorizes the location of food source and then starts exploiting it. After the foraging bee takes a part of nectar from the food source, it returns to the hive and saves the nectar to a food area in the hive. After saving the food, the bee enters to the decision making process and select one of the following options [3], [4]:

- If the nectar amount decreased to a low level or exhausted, she abandons the food source.
- If there are still sufficient amount of nectar in the food source, it can continue to forage without recruiting the nestmates.
- It can perform waggle dance to inform the nestmates about the same food source. After that it recruits the nestmates before the return to the food source.

A bee may select one of these options to switch its type based on different information such as quality of food source, its distance from the hive, its direction, and ease of extracting the food source [8].

The forager bees can memorize their historical information about the location and quality of food sources and use this information to make decisions at the next times intelligently. The bees which have this capability are called experienced foragers [3]. Using the historical information results a bee with more intelligence than forager bees. An experienced forager bee may obtain this intelligence using the information from waggle dance and use this information to select the next operation.

III. BEE SWARM OPTIMIZATION ALGORITHM

In this section we present the proposed bee swarm optimization algorithm (BSO). The BSO algorithm contains three types of bees: experienced forager, onlooker, and scout bees which fly in an *D*-dimensional search space $S \subset R^D$ to find the optimal solution. Assume that we have a set of bees in a swarm, β, these bees are partitioned as $\beta = \xi \cup \kappa \cup \vartheta$ based on their fitness, where ξ, κ, and ϑ respectively represent the sets of experienced forager, onlooker, and scout bees. In BSO algorithm, the fitness of a bee represents the quality of food source which is found by that bee so far. In this work, the percentage of scout, forager and experienced forager are determined manually. We use a small part of bees as scouts. The other part of bees is divided equally to onlooker and experienced forager bees (i.e. $n(\kappa) = n(\xi)$). Each bee i is associated with a position vector $\vec{x}(\beta, i) = (x(\beta, i1), x(\beta, i2), ..., x(\beta, iD))$, which represents a feasible solution for an optimal problem in the *D*-dimensional search space *S*. In BSO algorithm, each position vector $\vec{x}(\beta, i)$ represents a food source with an associated quality which is represented as $fit(\vec{x}(\beta, i))$.

The pseudocode of the BSO algorithm is represented in Fig 1. The BSO employs stochastic process to find optimal solution. Initially, number of the bees, $n(\beta)$, percentage of experienced forager, onlooker, and scout bees, and maximum number of iterations, $\text{Iter}_{\max}$, are determined. Also, other parameters are initialized. After that, at the start time of the algorithm all of the bees are positioned randomly in the search space:

$$\vec{x}_0(\beta, i) = Init(i, S) \quad \forall i \in \beta \tag{1}$$

where $Init(i, S)$ is the initialization function which associates a random position to the bee *i* in the search space *S*. After initialization, the bees of the swarm employ the following process to adjust their positions throughout iterations until the termination condition is met. At each iteration of the algorithm, the fitness of each bee is calculated and they are sorted based on their fitness values using the $Sort()$ function. Thereafter, each of the bees is classified as experienced forager, onlooker or scout. A predefined percentage of the bees which have the worst fitness are selected as scouts, while the remaining bees are divided equally as experienced foragers or onlookers. The first half of these bees which have better fitness is selected as experienced foragers and the other half are selected as employers. Classification of bees into the three categories provides a swarm with highly dynamic behavior which can use different flying patterns. The experienced forager, onlooker, and scout bees use these flying patterns to probabilistically adjust their trajectories in the search space for finding new food sources with better nectars.

In BSO algorithm, the food sources with poor qualities are abandoned and replaced by the new food sources which are found by the scout bees. The scout bees employ a random flying pattern to discover new food source and replacing the abandoned one with the new food source. A scout bee walks randomly in the search space and updates the position of the food source if the new food source has better quality than previously found food source by that bee. The random walk is performed by the scout bee in a region with radius τ. The search region is centered at

current position of the scout bee. So the next position of a scout bee is updated using the following equation:

$$\vec{x}_{new}(\vartheta,i)=\vec{x}_{old}(\vartheta,i)+Rw(\tau,\vec{x}_{old}(\vartheta,i)) \quad (2)$$

where $\vec{x}_{old}(\vartheta,i)$ represents the position of the abandoned food source which is replaced by the new food source positioned at $\vec{x}_{new}(\vartheta,i)$, and Rw is a random walk function that depends on the current position of the scout bee and the radius search τ. The initial value of radius τ is defined as a percentage of $|X_{\max}-X_{\min}|$, where $X_{\max}$ and $X_{\min}$ respectively represent the maximum and minimum values of the search space along a dimension. The value of τ is linearly decreased from $\tau_{\max}$ to $\tau_{\min}$ throughout iterations. The scout bees adjust their walking based on the τ. The large value of τ at the first iterations enables scout bees walking with large step size to explore wide regions in the search space. While the small values of τ in the last iterations encourage the scout bees to walk more precisely within small regions.

Initialization:

Determine $n(\beta)$, percentage of bees, *D*, and τ

For *i* = 1 **to** $n(\beta)$

$\vec{x}_{init}(\beta,i)=Init(i,S)$

Next *i*

Do

Compute fitness of the bees

Sort bees based on their fitness

Partition the swarm into the experienced forager, ξ, onlooker, κ, and scout, ϑ,bees

For *i* = 1 **to** $n(\xi)$

if $fit(\vec{x}(\xi,i))>fit(\vec{b}(\xi,i))$ **then** $\vec{b}(\xi,i)=\vec{x}(\xi,i)$

if $fit(\vec{b}(\xi,i))>fit(\vec{e}(\xi,\bullet))$ **then** $\vec{e}(\xi,\bullet)=\vec{b}(\xi,i)$

Next *i*

For *i* = 1 **to** $n(\xi)$

For *d* = 1 **to** D

$x_{new}(\xi,id)=x_{old}(\xi,id)+\omega_b r_b(b(\xi,id)-x_{old}(\xi,id))+\omega_e r_e(e(\xi,\bullet d)-x_{old}(\xi,id))$

Next *d*

Next *i*

For *i* = 1 **to** $n(\kappa)$

Select elite bee

For *d* = 1 **to** D

$x_{new}(\kappa,id)=x_{new}(\kappa,id)+\omega_e r_e(e(\xi,id)-x_{old}(\kappa,id))$

Next *d*

Next *i*

For *i* = 1 **to** $n(\vartheta)$

For *d* = 1 **to** D

$x_{new}(\vartheta,id)=x_{old}(\vartheta,id)+Rw(\tau,x_{old}(\vartheta,id))$

Next *d*

Next *i*

Adjust radius τ and step size *s*

Until termination condition is met

Fig 1. Pseudocode of bee swarm optimization algorithm

The success of an optimization algorithm highly depends on the balancing mechanism between exploration and exploitation. As mentioned in section 2, poor balance between exploitation and exploration results a weak algorithm. In previous work on bee algorithms such as [8]-[11], there exists a hard constriction on the trajectories of bees (e.g. the forager bees gravitates only to the elite bee). This may results the premature convergence. To cope with this problem, the BSO employs the experienced foragers that use their historical information about the food sources and their qualities. The information which is provided for an experienced forager bee is based on its own experience (or cognitive knowledge) and the knowledge of other experienced forager bees in the swarm. The cognitive knowledge is provided by the bees which memorize the decisions that they have made so far and the success of their decisions. An experienced forager *i* remembers the position of the best food source, denoted as $\vec{b}(\xi,i)=(b(\xi,i1),b(\xi,i2),...,b(\xi,i\mathrm{D}))$, and its quality which is found by that bee at previous times. The position of the best food source is replaced by the position of the new food source if it has better fitness (i.e. $\vec{b}(\xi,i)=\vec{x}(\xi,i)$ *if* $(fit(\vec{x}(\xi,i))>fit(\vec{b}(\xi,i)))$). The social knowledge is provided by sharing the nectar information of the food sources which are found by experienced forager bees. All the experienced forager bees select the best food source which is found by the elite bee as their interesting area in the search space. The position of the best food source is represented as the vector $\vec{e}(\xi,\bullet)=(e(\xi,\bullet1),e(\xi,\bullet2),...,e(\xi,\bullet\mathrm{D}))$. The elite bee is selected as an experienced forager with the highest fitness (i.e. $fit(\vec{e}(\xi,\bullet))>fit(\vec{b}(\xi,i))\ \forall i\in\xi$). Since the relative importance of cognitive and social knowledge can vary from one iteration to another, random parameters are associated to each component of position update equation. So, the position of an experienced forager bee $i\in\xi$ is updated using the following equation:

$$\vec{x}_{new}(\xi,i)=\vec{x}_{old}(\xi,i)+\omega_b r_b(\vec{b}(\xi,i)-\vec{x}_{old}(\xi,i))+\omega_e r_e(\vec{e}(\xi,\bullet)-\vec{x}_{old}(\xi,i)) \quad (3)$$

where r_b and r_e are random variables of uniform distribution in range of [0, 1] which model the stochasticity of flying pattern, the parameters ω_b and ω_e respectively control the importance of the best food source ever found by the *i*-th bee and the best food source which is found by elite bee, and $\vec{x}_{new}(\xi,i)$ and $\vec{x}_{old}(\xi,i)$ respectively represents the position vectors of the new and old food sources found by the experienced forager $i\in\xi$. The second component in the right side of the position update equation is the cognitive knowledge which represents that an experienced forager is attracted towards the best position ever found by that bee. The third component is the social knowledge which represents that an experienced forager is

attracted towards the best position $\vec{e}(\xi,\bullet)$ which is found by the interesting elite bee. The aforementioned flying pattern extends the movement trajectory of an experienced forager bee from a line to an area as shown in [11], and improves their searching abilities.

An onlooker bee uses the social knowledge provided by experienced forager bees to adjust its moving trajectory in the next time. At each cycle of the algorithm, the nectar information about food sources and their positions (social knowledge) which are provided by the experienced forager bees are shared in the dance area. After that, an onlooker bee evaluates the provided nectar information, employs a probabilistic approach to choose one of that food sources and follows the experienced forager bee which found the selected food source. In other word, an onlooker bee *i* selects an experienced forager *j* from set ξ as its own interesting elite bee, denoted as $\vec{e}(\xi,i)=(e(\xi,i1),e(\xi,i2),...,e(\xi,i\mathrm{D}))$, with probability p_j. The probability p_j is defined as a relative fitness of the selected experienced forager *j* in the set ξ:

$$p_j=\frac{fit(\vec{x}(\xi,j))}{\sum_{c=1}^{n(\xi)}fit(\vec{x}(\xi,c))} \tag{4}$$

where $fit(\vec{x}(\xi,i))$ is the fitness value of the food source which is found by the experienced forager bee *j* which is proportional to the quality of food source, and $n(\xi)$ is the number of experienced forager bees. The roulette wheel approach is used by onlooker bees for selecting their interesting elite bees. In this approach, as the quality of a food source is increased, the probability of its selection is increased, too. The flying trajectory of an onlooker bee *i* is controlled using the following equation:

$$\vec{x}_{new}(\kappa,i)=\vec{x}_{old}(\kappa,i)+\omega_e r_e(\vec{e}(\xi,i)-\vec{x}_{old}(\kappa,i)) \tag{5}$$

where $\vec{e}(\xi,i)$ is the position vector of the interesting elite bee for onlooker bee $i\in\kappa$ which is selected using (4), $\vec{x}_{old}(\kappa,i)$ and $\vec{x}_{new}(\kappa,i)$ respectively represents the position of the old food source and the new one which are selected by the onlooker bee *i*, and $w_e r_e$ probabilistically controls the attraction of the onlooker bee towards its interesting food source area.

The BSO algorithm provides a swarm of bees with different flying patterns. This heterogeneity result an effective population based algorithm for optimizing numerical functions. The random pattern is used by the scout bees. The BSO algorithm employs the scout bees to control diversity and stochasticity of the behaviors of the bees in the swarm. Usually, increasing diversity of population is used as a way to mitigate stagnation problem. In BSO algorithm, diversity of bee swarm is effectively controlled by adjusting the percentage of scout bees. Employing scout bees in an efficient way may produce valuable results. Re-initialization of scout bees is a one way which is performed to maintain diversity of population [11]. Re-initialization is a good choice in first iterations of the algorithm. In the last iterations, the algorithm tends to converge to optimum position, in such case, stagnation may occur and it seems that re-initialization is not useful. The BSO algorithm encourages the scout bees to fly randomly over the local areas to achieve better positions. The proposed approach permanently encourages the population to exit from stagnation state by exploring the nearby regions using scout bees. This approach provides excellent result in cases when the gradient does not point toward optimum solution or multiple local optima exist in direction of global optimum solution. The probabilistic pattern is used by the onlooker bees. The onlooker bees utilize the social knowledge about the food sources and their positions and probabilistically select one of them. This pattern encourages the swarm to efficiently control the exploration and convergence towards global optimum. Usually, premature convergence and stagnation may occur due to hard constriction on the movement trajectories of the individuals of a population. The third flying pattern which is used by experienced forager bees alleviates these problems. In our method, experienced foragers memorize the information of the previously found food sources and utilize this information to improve diversity of the algorithm.

IV. EXPERIMENTS

In this section, the experiments that have been done to evaluate the performance of the proposed BSO algorithm for a number of analytical benchmark functions are described. The previous studies on optimizations algorithms based on foraging behavior of the honey bees show that these algorithms have better performance in comparison with other population based algorithms such as genetic algorithms and PSOs [8]-[11]. So, we only focus on optimization algorithm inspired from foraging behavior of honey bees in this study. There exists few numerical function optimization algorithms based on the bee colony concepts. However, the performance of BSO algorithm is evaluated in comparison with two other bee algorithms. The comparisons are carried out in the case of numerical functions. These functions have been extensively used to compare optimization algorithms [8]-[11].

A. *Benchmarks*

To test the performance of BSO, six well known benchmark functions are used here for comparison, both in terms of optimum solution after a predefined number of iterations, and the rate of convergence to the optimum solution. TABLE I gives the test functions, mathematical expression, and their ranges. All test functions have optimum values equal to zero. To test the robustness of the algorithms, the most common initialization ranges used in the literature for these benchmarks considered in this paper. This commonly accepted approach, called asymmetric

initialization, is used in this paper. In asymmetric initialization the individuals of a population are initiated only in the right side of search, while the symmetric initialization employs the entire search space. Considering this approach, each bee has been initialized with a position which is randomly chosen in range $\left[\frac{X_{\max}}{2}, X_{\max}\right]$.

TABLE I
THE BENCHMARK FUNCTIONS, THEIR RANGES, OPTIMUMS, AND MAXIMUM VALUES FOR POSITION AND VELOCITY

Func.	Formula	Range
f_{Sph}	$f_{Sph}(x)=\sum_{i=1}^{n} x_i^2$	[-100, 100]
f_{Ros}	$f_{Ros}(x)=\sum_{i=1}^{n-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right)$	[-30, 30]
f_{Ras}	$f_{Ras}(x)=\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	[-5.12, 5.12]
f_{Shf6}	$f_{schf\,6}=0.5-\frac{\left(\sin\sqrt{\left(x_1^2+x_2^2\right)}\right)^2-0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2}$	[-100, 100]
f_{Gri}	$f_{Gri}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]
f_{Ack}	$f_{Ack}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{1}^{n}\cos(2\pi x_i)\right)+20+e$	[-30, 30]

B. Settings of the Algorithms

Simulations were performed to observe the performance of the proposed algorithm for finding optimum solutions. There are few algorithms in the literatures for numerical optimization based on foraging behaviors of honey bees. The performance of the new method (BSO) is compared with the performance of the Artificial Bee Colony (ABC) [8], and Bee and Foraging Algorithm (BFA) [10] methods. These algorithms have a few parameters; part of them is common while another part is specific for each algorithm.

Common parameters are number of dimensions of the search space, maximum generation, population size, and total number of trials. For all test functions with exception of 2D function Schaffer's f6, three different dimension sizes, 10, 20 and 30 are tested. The corresponding maximum generations are 5000, 7500 and 10000 respectively. For Schaffer's f6 function, the maximum generation is set to 1000. For all the test functions, the population size is set to 200, and a total of 100 trials for each experimental setting are conducted.

In BFA, parameter λ (distance of flight) has key role. We set $\lambda=11$ for all test functions. The parameter p_t which controls the selection of elite bee or other forger bees is set to 0.8. The percentage of onlooker and scout bees is controlled by parameter p_t throughout iterations. The onlooker bees are gravitated toward elite bee.

The same settings as presented in [8] and [11] are used in this paper for ABC algorithm. The percentage of onlooker and employed bees are equally set to 50% of the colony and the number of scout bees is selected as one.

In BSO algorithm, The percentage of onlooker bees is 48% of the swarm, the experienced forager bees is 48% of the swarm, and the scout bees is 4% of the swarm.

C. Experimental Results

In the experiments the number of iterations to reach a predefined threshold was specified for each function. Different success criteria for different functions are presented in the literature. Different success criteria for different functions are presented in the literatures. For Schaffer's f6, the success criterion is set to 0.00001, whereas for the other functions, the stopping criteria are set to 0.0001. After the final iteration, if the minimum value was reached by the algorithm was not below the threshold, the trial was considered unsuccessful. The values of mean, standard deviations, success rate, and the average number of iteration before the algorithms reach the success criteria in terms of each test function which are obtained by the BFA, ABC, and BSO algorithms are given in TABLE II. The X sign shows that the corresponding algorithm was failed in reaching success criteria in all trials. In TABLE II, the average fitness of the algorithms which are smaller than E-45 are considered as 0.

First, the performances of the algorithms are considered in terms of average optimum values for 100 trials. The results show that all of the algorithms provide good performance for Sphere and Schaffer's f6; however BSO and ABC strongly produce better results than BFA. It is clear from the result that for Rosenbrock function, the reduction of the quality of the average optimum was occurred for BFA and ABC compared with BSO. As described previously the BSO algorithm employs different flying patterns which help the swarm to maintain global search and control convergence towards global optima. These capabilities help the algorithm to cope with the smooth slope of the Rosenbrock function. For all the benchmark functions, the BSO algorithm outperforms BFA and ABC methods. This results show that the BSO produces consistent results for unimodal, multimodal functions with dependent or independent variables. The success rate which is related to the convergence of each method to the stopping criteria represents the stability of the algorithms. It is clear from the results that all the methods have converged to the stopping criteria for sphere function. The BFA algorithm has poor success rates on other test functions. The BFA algorithm only achieved 100% success rate on Sphere function in 10 dimension and Schaffer' f6 function. This algorithm fails in reaching success criteria for Rosenbrock and Griewank functions in 10, 20, and 30 dimensions as well as Rastrigrin function in 30 dimensions.

TABLE II
AVERAGE FITNESS VALUES, STANDARD DEVIATION, AND SUCCESS RATE OF THE ALGORITHMS FOR TESTUNCTIONS

			BFA				ABC				BSO			
Func.	Dim.	Max Iter	Avg	Stdv	Iter	S. R.	Avg	Stdv	Iter	S. R	Avg	Stdv	Iter	S. R
f_{Sph}	10	5000	0.000031	0.00024	196.9	**1**	4.926E-28	2.187E-25	124.6	**1**	**0.00**	0.00	**95.7**	**1**
	20	7500	0.000892	0.00103	423.8	0.73	3.071E-25	5.368E-23	387.0	**1**	**0.00**	0.00	**313.2**	**1**
	30	10000	0.093786	0.02745	783.4	0.21	6.355E-24	1.240E-22	556.2	**1**	**0.00**	0.00	**490.6**	**1**
f_{Ros}	10	5000	7.2084513	9.436551	X	0.00	0.009252	0.010890	X	0.00	**3.617E-7**	1.081E-6	**549.3**	**1**
	20	7500	13.927110	16.830792	X	0.00	0.012870	0.017322	X	0.00	**9.201E-7**	5.102E-6	**804.5**	**1**
	30	10000	21.276334	25.03812	X	0.00	0.034010	0.048012	X	0.00	**5.865E-6**	8.519E-5	**1225.2**	**1**
f_{Ras}	10	5000	0.003821	0.006513	3693.4	0.07	4.398E-24	6.195E-23	498.6	**1**	**0.00**	0.00	**487.1**	**1**
	20	7500	0.017613	0.100691	3782.5	0.03	2.011E-22	7.048E-21	1109.3	**1**	**0.00**	0.00	**946.4**	**1**
	30	10000	0.967820	4.24561	X	0.00	8.609E-22	5.332E-21	1590.4	**1**	**0.00**	0.00	**1243.8**	**1**
f_{Ack}	10	5000	0.000085	0.000237	2725.8	0.35	8.462E-12	9.307E-12	461.2	**1**	**7.105E-19**	5.482E-18	**207.6**	**1**
	20	7500	0.000639	0.003406	3137.4	0.14	3.285E-13	1.034E-12	703.5	**1**	**2.131E-19**	4.603E-18	**343.2**	**1**
	30	10000	0.001398	0.002054	3485.2	0.08	7.632E-13	2.760E-12	942.6	**1**	**1.460E-18**	8.130E-17	**438.9**	**1**
f_{Gri}	10	5000	3.209850	4.298031	X	0.00	0.000219	0.000691	478.0	0.47	**0.00**	0.00	**431.8**	**1**
	20	7500	1.792011	1.973611	X	0.00	3.170E-9	1.725E-7	1031.3	**1**	**0.00**	0.00	**515.3**	**1**
	30	10000	0.981425	1.023101	X	0.00	5.578E-10	3.008E-8	1485.8	**1**	**0.00**	0.00	**611.4**	**1**
f_{Shf6}	2	2000	9.381E-8	1.0348E-7	198.3	1	3.806E-16	3.183E-14	126.7	**1**	**0.00**	0.00	**93.0**	**1**

Its success rates vary from 3% to 73% for other test functions in different dimensions. This problem occurs due inability of BFA in providing proper balance between exploration and exploitation. All of the bees in BFA except scout bees select the elite bee as their interesting dancer and adjust their trajectories toward the elite bee without considering other valuable information. So, they are gravitated rapidly toward the elite bee and the premature convergence will be occurred. The ABC algorithm achieves a success rate 100% on Sphere, Rastrigrin, Ackley, and Schaffer's f6 function in 10, 20, and 30 dimensions. While ABC algorithms have good success rate in most of the test functions, their success rate drastically decrease in the case of Rosenbrock function. This is caused due to very smooth slope of the Rosenbrock function nearby its global optimum position. Further, for the Griewank function in 10 dimensions, only 47 out of 100 trials from the ABC method have converged to stopping criteria. It is clear from the success rate results that the BSO method produces stable solutions for solving numerical optimization problems. The success rate of 100% on all the test functions in 10, 20, and 30 dimensions were reached by BSO algorithm.

V. CONCLUSIONS

The optimization algorithms which are inspired from intelligent behavior of honey bees are among the most recently introduced population based algorithms. In this paper, we have described a novel optimization algorithm, called BSO, based on foraging behavior of honey bees. Different types of flying patters were introduced into the BSO algorithm aiming to maintain proper balance between global and local search by providing diversity into the swarm of bees. The BSO algorithm was compared with two other bee colony optimization algorithms on the six benchmark functions. The experiments results showed that the BSO and is effective and powerful algorithm for numerical function optimization and outperform other algorithms investigated in this study.

REFERENCES

[1] Melissa D. Cox, Mary R. Myerscough, "A flexible model of foraging by a honey bee colony: the effects of individual behaviour on foraging success", in *Journal of Theoretical Biology*, 223, pp. 179–197, 2003.

[2] N. Karaboga, "A new design method based on artificial bee colony algorithm for digital IIR filters", *Journal of the Franklin Institute* 346, pp. 328–348, 2009.

[3] A. Baykasoglu, L. Ozbakır, and P. Tapkan, *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, 2007.

[4] D. Teodorovic, M. Dell Orco, "Bee Colony Optimization – A Cooperative Learning Approach to Complex Transportation Problems", in *Journal of Advanced OR and AI Methods in Transportation*, pp. 51-60, 2007.

[5] Dusan Teodorovic, Panta Lucic, Goran Markovic, Mauro Dell' Orco, "Bee Colony Optimization: Principles and Applications", in *proceeding of 8th Seminar on Neural Network Applications in Electrical Engineering*, Neurel 2006, pp. 151-156.

[6] D.T. Pham, M. Castellani, and A. A. Fahmy, "Learning the Inverse Kinematics of a Robot Manipulator using the Bees Algorithm", *The IEEE International Conference on Industrial Informatics*, pp. 493-498, 2008.

[7] Hesham Awadh A. Bahamish, Rosni Abdullah, Rosalina Abdul Salam, "Protein Conformational Search Using Bees Algorithm", in *Proceeding of Second Asia International Conference on Modelling & Simulation*, pp. 238-244, 2005.

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

[9] Yang, X.S.: Engineering optimizations via nature-inspired virtual bee algorithms. *Lecture Notes in Computer Science*, pp. 317–323. Springer, GmbH (2005).

[10] K. Sundareswaran, and V.T. Sreedevi, "Development of Novel Optimization Procedure Based on Honey Bee Foraging Behavior", in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1220-1225, 2008.

[11] D. Karaboga, B. Akay, "A comparative study of Artificial Bee Colony algorithm", in *Journal of Applied Mathematics and Computation*, 2009.