Commun Nonlinear Sci Numer Simulat 15 (2010) 3142–3155

Contents lists available at ScienceDirect

Commun Nonlinear Sci Numer Simulat

journal homepage: www.elsevier.com/locate/cnsns

A novel bee swarm optimization algorithm for numerical function optimization

Reza Akbari *, Alireza Mohammadi, Koorush Ziarati

Department of Computer Science and Engineering, Shiraz University, Shiraz, Iran

ARTICLE INFO

Article history:
Received 12 June 2009
Received in revised form 2 November 2009
Accepted 2 November 2009
Available online 10 November 2009

Keywords:
Bee swarm optimization
Numerical function optimization
Time-varying weights
Repulsion factor

ABSTRACT

The optimization algorithms which are inspired from intelligent behavior of honey bees are among the most recently introduced population based techniques. In this paper, a novel algorithm called bee swarm optimization, or BSO, and its two extensions for improving its performance are 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. As the first extension, the BSO algorithm introduces different approaches such as repulsion factor and penalizing fitness (RP) to mitigate the stagnation problem. Second, to maintain efficiently the balance between exploration and exploitation, time-varying weights (TVW) are introduced into the BSO algorithm. The proposed algorithm (BSO) and its two extensions (BSO–RP and BSO–RPTVW) are compared with existing algorithms which are based on intelligent behavior of honey bees, on a set of well known numerical test functions. The experimental results show that the BSO algorithms are effective and robust; produce excellent results, and outperform other algorithms investigated in this consideration.

1. 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. So, unconstrained optimization problems can be formulated as a D-dimensional minimization or maximization problem:

$$\min(\text{or } \max) f(\vec{x}), \quad \vec{x} = (x_1, x_2, \ldots, x_D) \tag{1}$$

where D is the number of the parameters to be optimized. 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 success of a population based method depends on its ability to establish proper balance between exploration and exploitation. A poor balance between exploration and exploitation may result a weak optimization method which may suffer from premature convergence, trapping in a local optima, and stagnation.

GA is the most popular evolutionary algorithms, in which a population of individuals evolves according to a set of rules such as selection, crossover and mutation [1]. In such algorithm, exploitation is obtained through selection, where individ-

* Corresponding author. Tel.: +98 9171003767.
E-mail addresses: rakbari@cse.shirazu.ac.ir (R. Akbari), a_mohammadi@cse.shirazu.ac.ir (A. Mohammadi), ziarati@shirazu.ac.ir (K. Ziarati).

1007-5704/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cnsns.2009.11.003

uals move toward regions with higher fitness. Also, the exploration is provided by perturbing individuals using crossover and mutation operators [2].

PSO is a swarm intelligence technique developed by Eberhart and Kennedy [3], inspired by the social behavior of bird flocking and fish schooling. PSO has been shown successfully optimizes a wide range of continuous functions. The search for an optimum in PSO is an iterative process that is based on random decisions. In PSOs, exploitation is obtained through selecting the particle with best fitness as *gbest* and moving toward that particle. Each particle memorizes its previous best position, represented as *pbest* to explore the space between its previous best position and global best position. Although, PSO provides powerful methods for optimization problems, it suffers from different problems such as premature convergence and stagnation. Different approaches such as inertia weight, and time-varying coefficients were proposed to alleviate these problems [4–7].

The algorithms which are inspired from intelligent behaviors of honey bees constitute third class of population based methods. These algorithms have been developed and applied to different engineering fields [8–14]. 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 [15]. 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 [16]. 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 represents the best possible solution [17]. 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 the 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 [15–19].

Similar to other swarm based optimization algorithms, it is important to establish a proper balance between exploration and exploitation in bee swarm optimization approaches. In a bee swarm, different behaviors of the bees provide this possibility to establish powerful balancing mechanism between exploration and exploitation. This property provides the opportunity to design more efficient algorithms in comparison with other population based algorithms such as PSO and GA.

It seems that exploration of the search space is achieved through two different ways. The first way is provided by scout bees. Spontaneous searching of the scout bees navigate them to explore new regions beyond that defined by employer or onlooker bees. Patterns by which the foragers and onlookers control their movements may provide good exploration approach. For example experienced foragers can use historical information about location and quality of food source to adjust their movement patterns in the search space. This approach alleviates the hard constriction on the search trajectory of a forager bee and extends the search trajectory to a search area. Usually, in bee swarms, an exploitation mechanism is based on two different behaviors. In first way, a honey bee advertise its' food source by dancing in dance area. This behavior encourages other bees to select a dancer bee and follow her. In the second way, a forager try to forage its' food source without advertising it.

This work presents a novel method based on foraging behaviors of honey bees. 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 memorizes the information of the best food source which is found so far by it. It selects the best experienced forager bee as the elite bee, and adjusts its position based on the cognitive and social knowledge. The BSO algorithm utilizes a set of approaches to mitigate the stagnation and premature convergence problems. These approaches include repulsion factor, penalizing fitness, random walking, time-varying weights, and alleviating the hard constriction on the moving direction of the bees.

The paper is organized as follows. Section 2 introduces the intelligent behaviors of honey bees. Description of the proposed BSO algorithm along with approaches for mitigating of stagnation and premature convergence are presented in Section 3. Section 4 reports numerical test functions, experimental settings of the algorithms, and experimental analysis on the proposed approach in comparison with other algorithms. Finally, Section 5 concludes this work.

2. Intelligent behaviors of honey bees

A swarm 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 [12]. One of the main differences of the bee swarm in comparison with other population based methods is that it contains different types of bees such as scouts, onlookers, foragers, etc. Therefore, a bee swarm can provide different types of patterns which are used by the bees to adjust their flying trajectories. Therefore, we expect that this capability result effective and robust algorithms to solve highly complex problems.

A bee may have one of the following types: (employed/unemployed) forager, scout, onlooker, recruit, and experienced forager [10–13]. The type of a bee depends on the action which is performed and the level of information which may be used

by it. A potential forager will start as unemployed forager. This type of bee 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. Scout bees fly spontaneously around the hive and search for new food sources without any knowledge about the environment. Usually, small part of a swarm is selected as scout bees. It is possible to adjust the number of scout bees adaptively according to the gathered information from the environment. The 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 the 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 three options (type of the bee at the next time depends on its selection) [10,11]:

- If the nectar amount decreased to a low level or exhausted, she abandons the food source (the forager bee becomes to unemployed forager).
- If there are still sufficient amount of nectar in the food source, it can continue to forage without recruiting the nestmates (the bee remains its type as forager).
- It can perform waggle dance to inform the nestmates about the same food source. After that it recruits the nestmates before she returns to the food source (the bee remains its type as forager).

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. For numerical optimization, the food source is considered as a point in the search space, and its' quality corresponds to the fitness of the point [15].

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 [10]. 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. For example, the bee may try to explore a food source if there are some other bees confirm the quality of that food source; she may adjust its flying trajectory based on information of the previously found food source and the new food sources, or she can be scout bee to search new areas if the fitness of the food source decreases rapidly.

3. 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 a D-dimensional search space $S \subset R^D$ to find the optimum solution. Assume that we have a set of bees in a swarm which represented as β. As shown in Fig. 1, these bees are partitioned as $\beta = \vartheta \cup \kappa \cup \xi$ 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 $\vec{x}(\beta, i) = (x(\beta, i1), x(\beta, i2), \ldots, x(\beta, iD))$, 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 presented in Fig. 2. 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 the bees are positioned randomly in the search space:

$$\vec{x}_0(\beta, i) = Init(i, S) \quad \forall i \in \beta \tag{2}$$

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

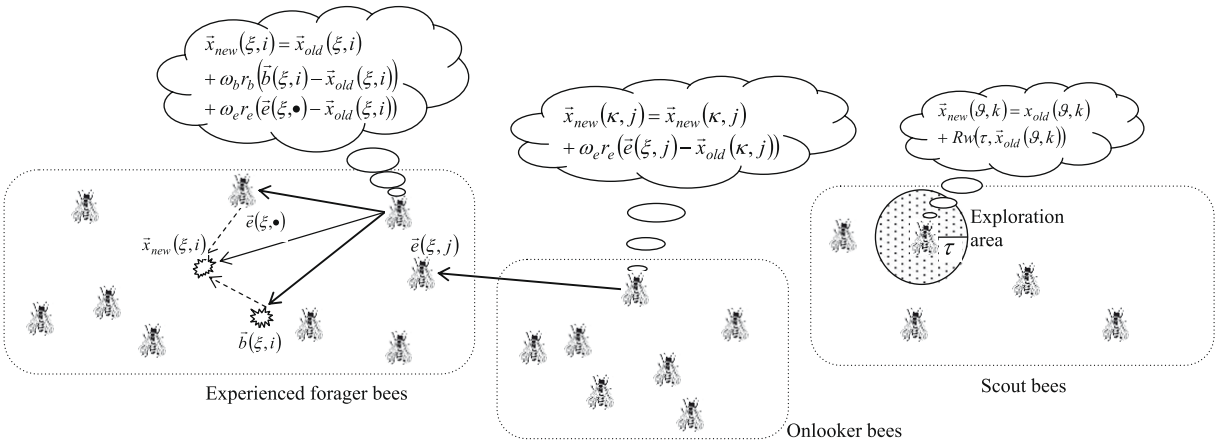

Fig. 1. The structure of the bee swarm and the flying patterns of the bees: the scout bees perform random walk around their current positions. An onlooker bee selects an experienced forager probabilistically as its interesting elite bee and follows it. The experienced forager bees memorize their historical information, select the global best bee as the elite bee and update their position according social and cognitive knowledge.

Initialization:

Determine the number of bees, $n(\beta)$, percentage of experienced foragers, onlookers, and scouts, the dimension, n, radius τ, and termination condition, $\text{Iter}_{\max}$.

For $i = 1$ **to** $n(\beta)$ //do for all the bees

$\vec{x}_{init}(\beta,i)=Init(i,S)$ //initialize the bees randomly in the search space 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** ξ //do for all the experienced forager bees

if $fit(\vec{x}(\xi,i))>fit(\vec{b}(\xi,i))$ **then** $\vec{b}(\xi,i)=\vec{x}(\xi,i)$ //update the previous best position

if $fit(\vec{b}(\xi,i))>fit(\vec{e}(\xi,\bullet))$ **then** $\vec{e}(\xi,\bullet)=\vec{b}(\xi,i)$ //select elite bee for all the experienced forager bees

Next i

For $i = 1$ **to** ξ //do for each experienced forager bee

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

$x_{new}(\xi,id)=x_{old}(\xi,id)+\omega_b r_b(b(\xi,id)-x_{old}(\xi,id))+\omega_e r_e(e(\xi,\bullet d)-x_{old}(\xi,id))$ //update the position of an experienced forager bee i

Next d

Next i

For $i = 1$ **to** κ //do for each onlooker bee

Select elite bee //select an elite bee from experienced forager bee foe onlooker i

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

$x_{new}(\kappa,id)=x_{new}(\kappa,id)+\omega_e r_e(e(\xi,id)-x_{old}(\kappa,id))$ //update the position of an onlooker bee i

Next d

Next i

For $i = 1$ **to** ϑ //do for each scout bee

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

$x_{new}(\vartheta,id)=x_{old}(\vartheta,id)+Rw(\tau,x_{old}(\vartheta,id))$ //walk randomly around search space

Next d

Next i

Adjust radius τ and step size s //adjust parameters of the search space for scout bees

Until termination condition is met

Fig. 2. Pseudocode of bee swarm optimization algorithm

divided equally as experienced foragers and 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 flying patterns are presented in Fig. 1. 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 scout bee walks randomly 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:

$$\vec{x}_{new}(\vartheta, i) = \vec{x}_{old}(\vartheta, i) + Rw(\tau, \vec{x}_{old}(\vartheta, i)) \tag{3}$$

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 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.

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 [15–19], there exists a hard constriction on the trajectories of bees (e.g. the forager bees gravitates only to the elite bee). This may result the premature convergence. To cope with this problem, 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 experienced foragers. These bees 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 $\vec{b}(\xi, i) = (b(\xi, i1), b(\xi, i2), \ldots, b(\xi, iD))$, 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 $\vec{e}(\xi, \bullet) = (e(\xi, \bullet 1), e(\xi, \bullet 2), \ldots, e(\xi, \bullet D))$. The elite bee is selected as an experienced forager with the highest fitness (i.e. $fit\bar{e}\xi, (\bullet) > fit(\vec{b}(\xi, i))\ \forall i \in \xi$). Since the relative importance of cognitive and social knowledge can vary from one cycle 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:

$$\vec{x}_{new}(\xi, i) = \vec{x}_{old}(\xi, i) + \omega_b r_b(\vec{b}(\xi, i) - \vec{x}_{old}(\xi, i)) + \omega_e r_e(\vec{e}(\xi, \bullet) - \vec{x}_{old}(\xi, i)) \tag{4}$$

where r_b and r_e are random variables of uniform distribution in range of [0, 1] which model the stochasticity of the flying pattern, the parameters ω_b and ω_e, respectively control the importance of the best food source ever found by the ith 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 which are 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 Fig. 1, and improves their searching abilities.

An onlooker bee uses the social knowledge provided by the experienced forager bees to adjust its moving trajectory at 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 the 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 $\vec{e}(\xi, i) = (e(\xi, i1), e(\xi, i2), \ldots, e(\xi, iD))$, with probability p_j. The probability p_j is defined as a relative fitness of the selected experienced forager j in the set ξ:

$$p_j = \frac{fit(\vec{x}(\xi, j))}{\sum_{c=1}^{n(\xi)} fit(\vec{x}(\xi, c))} \tag{5}$$

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:

$$\vec{x}_{new}(\kappa, i) = \vec{x}_{old}(\kappa, i) + \omega_e r_e(\vec{e}(\xi, i) - \vec{x}_{old}(\kappa, i)) \tag{6}$$

where $\vec{e}(\xi, i)$ is the position vector of the interesting elite bee for onlooker bee $i \in \kappa$ which is selected using (5), $\vec{x}_{old}(\kappa, i)$ 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 results 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 [19]. 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 control efficiently 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

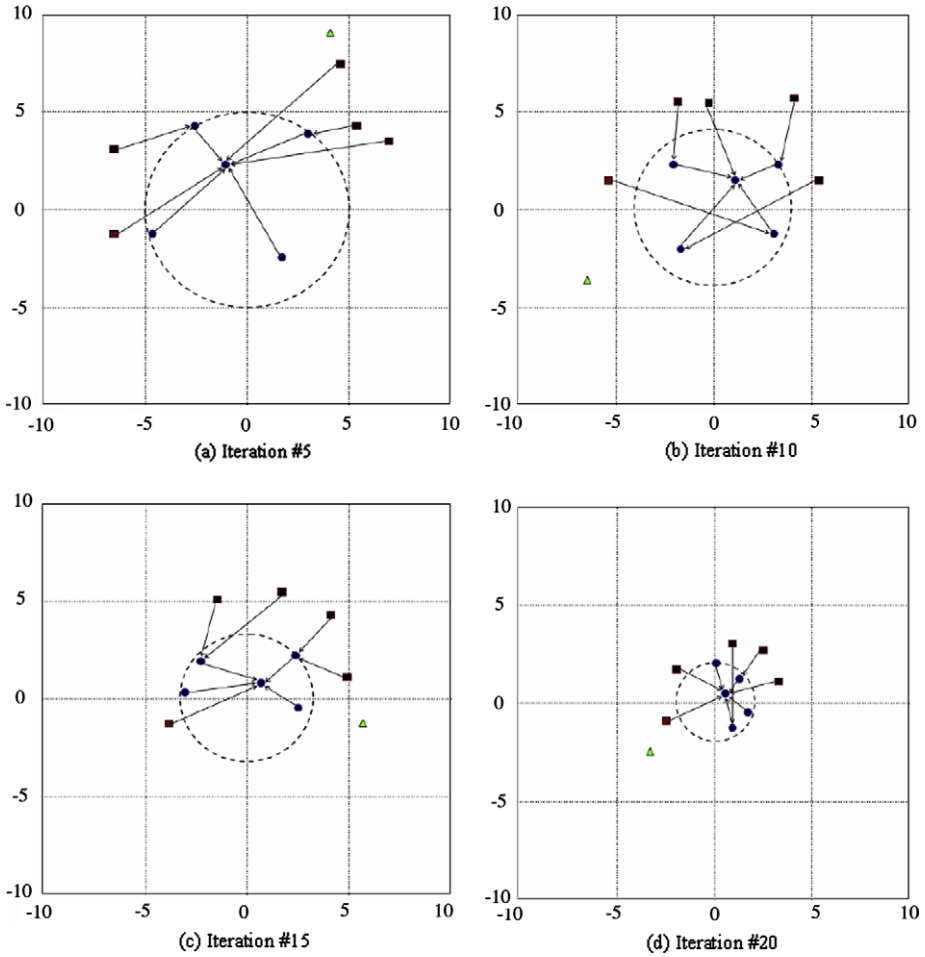

Fig. 3. Distribution of different types of bees on the two dimension search space. Triangle, circles and squares, respectively represent scout, experienced forager and onlooker bees.

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 the diversity of the algorithm.

As a convenient observation, the convergence behavior of the BSO algorithm is presented. The convergence of BSO algorithm for a group of $n = 11$ bees (5 experienced foragers, 5 onlookers, and 1 scout) for two dimensional Sphere function is shown in Fig. 3. Fig. 3(a)–(d) presents the distribution of bees with different roles at 5th, 10th, 15th, and 20th iterations. The circles, squares, and triangle signs, respectively represent experienced forager, onlooker, and scout bees. As described before, and shown in this figure we can see that the distribution of bees in the first iterations depicts that BSO algorithm prefer exploration, while shrinking bees in last iterations represents that BSO algorithm tends to exploit around local optima. This process is occurred due to stochastic behaviors of bees which modify topology of the swarm constantly. At each cycle of the algorithm, the worst bee is selected as scout bee. The experienced foragers and onlookers are selected based on their fitness. So, the onlooker and scout bees are placed at the border of the swarm while the experienced forager bees are placed inside the swarm (the dashed circle partition the swarm into the experienced forager and onlooker bees). The experienced forager bees select the best bee as their interesting elite bee, and onlooker bees stochastically select their interesting elites from the experienced forager bees which are placed at the interior region of the swarm. This process provides powerful way to maintain balance between exploration and exploitation. The onlooker and scout bee control the global search while the experienced forager bee maintain the convergence towards the global optimum.

3.1. Approaches to mitigate stagnation

Stagnation and premature convergence are the major weaknesses of the population based algorithms. As mentioned before, the flying patterns in BSO algorithm control diversity of the swarm and provide an effective way to balance between exploration and exploitation. This approach alleviates the premature convergence problem. An extension of the BSO algorithm which aims to alleviate stagnation problem and improve its performance is suggested in this section. The stagnation occurs when the algorithm reaches its convergence in which the stagnant individuals of the population choose the same position and the population traps to local optima. We propose the following approaches (repulsion factor and penalizing fitness) aiming to alleviate stagnation to improve the performance of the BSO algorithm in terms of minimizing numerical functions. We call the BSO algorithm with these extensions as BSO–RP.

Repulsion factor: The concept of repulsion technique was first introduced in PSO methods [20–22]. As stated by Eberhart [23], "the repulsion technique adds the ability to guarantee that all individuals of a population will not move toward the best solution which is found so far. Therefore, an algorithm with repulsion technique can have the ability to avoid already found solutions and, therefore, to have more chance to find new position of search space with better fitness". The repulsion technique provides this ability by increasing diversity of the population. To increase diversity, a repulsion technique encourages a part of individuals to move in opposite directions of their elites.

In bee colony algorithms, the onlooker bees move according to the past experience of forager bees. In this approach the new population inherits the experience of the past population and the individuals of population move toward the food sources which are found by forager bees according to their fitness. This procedure causes that the population ignores a part of search space that is behind of forager bees and may containing better solutions. So, the population may spend more time to explore this part of search space.

In order to alleviate this problem and enhance the searching ability of the bees in BSO–RP, we suggest a parameter called repulsion factor. The repulsion factor, represented as $sign(r)$, encourage a bee to modify its search direction and move toward a region which may ignore by the population. Based on these considerations, the position update equations for onlooker and experienced foragers are modified as follows:

$$\vec{x}_{new}(\kappa, i) = \vec{x}_{old}(\kappa, i) + sign(r)\{\omega_e r_e(\vec{e}(\xi, i) - \vec{x}_{old}(\kappa, i))\} \tag{7}$$

$$\vec{x}_{new}(\xi, i) = \vec{x}_{old}(\xi, i) + sign(r)\{\omega_b r_b(\vec{b}(\xi, i) - \vec{x}_{old}(\xi, i)) + \omega_e r_e(\vec{e}(\xi, \bullet) - \vec{x}_{old}(\xi, i))\} \tag{8}$$

where $sign(r)$ is defined as:

$$sign(r) = \begin{cases} 1 & \text{if } (r \leqslant P_{rf}) \\ -1 & \text{if } (r > P_{rf}) \end{cases} \tag{9}$$

where r is a random parameter which is chosen uniformly from the interval [0, 1], and P_{rf} is a predefined probability which controls the rate of repulsion in the swarm. The repulsion factor maintains the diversity of the population by means of P_{rf} parameter. As the value of P_{rf} increases the level of diversity in population decreases. As a consequence, this approach successfully extends the search range of the population.

Penalizing fitness: To further mitigate the stagnation problem we use a parameter called penalty. As stated by Parsapolous [24], the penalty parameter is used to escape from local optima. The penalty is assigned to the fitness of a solution. In BSO algorithm, the best solutions are maintained by experienced forager bees. At stagnation time, usually, a part of experienced forager bees trapped in local optima and other bees stop moving once they catch up with their interesting experienced forager bees. Therefore, we can use penalty parameter for the stagnant experienced forager bees to escape them from local op-

tima. Penalty is a function of solution age. Solution age is determined as a number of iterations in which the solution is not changed. The fitness of a stagnant bee i is decreased using the following equation:

$$fit(\vec{x}(\beta,i)) = \rho(a_i) \times fit(\vec{x}(\beta,i)) \tag{10}$$

where $\rho(a_i)$ represents the penalty function, and a_i is the age of the bee i. The penalty of a bee increases as its age increases. The penalty decreases the fitness of a bee which is trapped in a local optima, and encourages that bee to escape form its' position and gravitate toward other bees.

3.2. Time-varying weights

Another concept "time-varying weights" (TVW) is introduced here to further improve the performance of BSO algorithm. During the first stages of a swarm optimization method we prefer that individuals of the swarm wander through the entire search space without convergence toward local optima, while during the last stages, it is desirable to improve the moving patterns of individuals to enhance convergence toward the optimum solution. The optimization algorithms based on bee colonies provide more flexibility than other population based algorithms to control the wandering and converging mechanism respectively in the first and the last iterations. In one hand, the bee algorithms use scout bees which wander through entire search space throughout iteration. In other hand, the onlooker bees stochastically select the forager to follow them. This process provides relatively good exploration through search space. However, a drawback of the bee algorithm is that the onlooker bees are gravitated toward selected foragers rapidly. This may encourage the individuals to cluster around local optima and premature convergence occurs.

The aforementioned concerns encourage us to propose dynamic weights for the BSO algorithm (BSO–RPTVW). The dynamic weights aim to enhance the global search in the first iterations and encourage the bees to converge toward the optimum solution in the last iterations. This approach has been widely used in PSO methods [25,26]. To achieve this goal the dynamic weights are developed for (3) and (5). The Eqs. (3) and (5) show that, for experienced foragers, the search toward the optimum solution is guided by the two weight parameters (the cognitive component and the social component) while for onlookers, the search is guided only by social component. However, it seems that proper control of these two components is very important to find the optimum solution accurately and efficiently.

It seems that a relatively high value of the ω_b, in comparison with the weight ω_e, encourage the individuals to wander the search space, while a relatively high value of the ω_e may lead bees to prematurely converge toward a local optimum. Considering this property, we reduce the importance of the cognitive part and increase the importance of the social part. For this purpose, the forager bees start with a large cognitive part and small social part at the first iterations and end with a small cognitive part and a large social part. This allows the bees to move around the search space in the first iterations without converging toward local optima. Also, the bees are allowed to converge to the best solution in the last iterations. In our model the weights are dynamically adjusted through iterations. Normally, the value of ω_b is set to a high value and decreased and the value of ω_e is set to a low value and increased linearly through iterations to balance between individual knowledge and that of the others. The dynamics weights ω_b and ω_e are represented as the following formulations:

$$\omega_b = \omega_{b,\max} - \frac{\omega_{b,\max} - \omega_{b,\min}}{iter_{\max}} \times iter \tag{11}$$

$$\omega_e = \omega_{e,\min} + \frac{\omega_{e,\max} - \omega_{e,\min}}{iter_{\max}} \times iter \tag{12}$$

where $\omega_{b,\max}$ and $\omega_{e,\max}$, respectively represent the maximum values of the cognitive and social parts, $\omega_{b,\min}$ and $\omega_{e,\min}$, respectively represent the minimum value of the cognitive and social part, *iter* is the current iteration number, and $iter_{\max}$ is the maximum number of allowable iterations. In this paper, the weight ω_b changes linearly from 1.5 to 0.75 while the weight ω_e linearly increases from 0.75 to 1.5.

4. Experiments

In this section, the experiments that have been done to evaluate the performance of the proposed BSO algorithm and its two extensions 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 [15–19]. 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 algorithms based on artificial bees. The comparisons are carried out in the case of numerical continuous functions. The continuous test functions in this experiment have been extensively used to compare population based optimization algorithms.

Table 1
The benchmark functions, their rand, optimum value, and maximum values for position and velocity.

Function	Formula	Type	Range	Optimum position	Optimum value
f_{Sph}	$f_{Sph}(x)=\sum_{i=1}^{n}x_i^2$	Unimodal	[−100, 100]	(0,0,. . .,0)	0
f_{Ros}	$f_{Ros}(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	Unimodal	[−30, 30]	(1,1,. . .,1)	0
f_{Ras}	$f_{Ras}(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	Multimodal	[−5.12, 5.12]	(0,0,. . .,0)	0
f_{Shf6}	$f_{Schf6}=0.5-\frac{\left(\sin\sqrt{(x_1^2+x_2^2)}\right)^2-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	Multimodal	[−100, 100]	(0,0,. . .,0)	0
f_{Gri}	$f_{Gri}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	Multimodal	[−600, 600]	(0,0,. . .,0)	0
f_{Ack}	$f_{Ack}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_1^n x_i^2}\right)-\exp\left(\frac{1}{n}\sum_1^n\cos(2\pi x_i)\right)+20+e$	Multimodal	[−30, 30]	(0,0,. . .,0)	0

4.1. 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. These benchmarks are widely used in evaluating performance of population based methods [4–7]. The first two functions are unimodal, while others are multimodal. A function is called unimodal, if it has only one optimum position. The multimodal functions have two or more local optima.

Table 1 gives the test functions, mathematical expression, their optimum values, their optimum positions, and ranges. 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 the search space, while the symmetric initialization employs the entire search space (the works presented in [9,13] were used symmetric initialization). Considering this approach, each bee has been initialized with a position which is randomly chosen in range $\left[\frac{X_{max}}{2},X_{max}\right]$. To control the explosion of each bee, the excessive searching outside the predefined search space is limited. During a trial of each algorithm, the position values of a bee are limited to interval $[X_{min};X_{max}]$.

We select these test function as each of them is a candidate for a different class of real-world problems. They have different characteristics (e.g. they are unimodal or multimodal, or have dependent or independent variables). A robust optimization algorithm maintaining balance between exploration and exploitation, controlling diversity, mitigating premature convergence and stagnation to cope with problems of different types.

The Sphere has independent variables, contains no local optima, and have smooth gradient toward global optimum. It represents an easy problem which successfully solved by many population based optimization algorithm. The Rosenbrock function has smooth slope around its' global optimum position, its global optimum lays inside a long, narrow, and parabolic shaped flat valley, its variables are strongly dependent, and the gradient do not point towards its optimum position. All of these cause that the convergence toward the global optimum in Rosenbrock be relatively difficult. This function has been frequently used to test the optimization algorithms. The algorithms with hard constriction on movement trajectories of their individuals may easily encounter to stagnation. The Schaffer' f6 is a multimodal function with dependant variables. It has smooth slope around global optimum, so methods with poor flying patterns encounter problem in regions nearby global optimum. An optimization algorithm should increase diversity of the population to cope with the problem of this type.

Ackley's is a widely used multimodal test function. This function has one narrow global optimum basin and numerous local optima. In comparison with other multimodal functions, it represents a relatively easy problem as it has shallow local optima. There is no dependency between variables of the Rastrigrin test function. The cosine modulation produces frequent local optima. So, this test function is highly multimodal which makes it to be a complex problem. An optimization algorithm should provide an efficient balance between exploration and exploitation and have good diversity to overcome the problems of this type. The Griewank's function is based on the Sphere function. Similar to Rastrigrin function it has many wide spread local optima regularly distributed. Its second component represents linkage between variables which make it to a difficult multimodal problem. The local optima are located in direction of gradient, so an optimization algorithm should provide efficient balance between global and local search to solve this type of problem. The Griewank's function with high dimensionality seems unimodal.

4.2. 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) and its two extensions (BSO–RP and BSO–RPTVW) are compared with the performance of the Artificial Bee Colony (ABC) [15], and Bee and Foraging Algorithm (BFA) [17] 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 10,000, respectively. For Schaffer's f6 function, the maximum generation is set to 2000. 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, the parameter λ, the 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 [15,19] 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. In BSO–RP algorithm, the control parameter p_{rf} is set to 0.8, and the maximum and minimum values of the parameter τ are set as $\tau_{\max} = 1$ and $\tau_{\min} = 0.2$. In BSO–RPTVW algorithm, the maximum values of the dynamic weights are set 1.5 (i.e. $\omega_{b,\max} = \omega_{e,\max} = 1.5$), and their minimum values are set to 0.5 (i.e. $\omega_{b,\min} = \omega_{e,\min} = 0.5$). The value of ω_b linearly decreases from its' maximum value to its' minimum value, while the value of ω_e linearly increases from its' minimum value to its' maximum value.

4.3. 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 literatures. For Schaffer's f6, the success criterion was set to 0.00001, whereas for the other functions, the stopping criteria were 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 iterations before the algorithms reach the success criteria in terms of each test function which are obtained by the BFA, ABC, BSO, BSORP, and BSO–RPTVW algorithms are given in Tables 2 and 3.To ease of observation, the best results obtained by the algorithms are shown in bold. The X sign in Table 3 represents that the corresponding algorithm was failed in reaching success criteria in all trials. 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 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 and its two extensions. As described previously the BSO algorithm and its two extensions employ different flying patterns which help the swarm to maintain global search and to 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 and its two extensions outperform BFA and ABC methods. The BSO–RP and BSO–RPTVW produce better results in terms of optimum solution than BSO algorithm. This is occurred, due to using repulsion factor and penalizing fitness as well as time-varying weights which maintain the diversity of algorithm and control the global and local searches regardless of the type of the considered function. The results show that the BSO and its extensions produce 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. 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 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 and its two extensions produce 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, BSO–RP, and BSO–RPTVW algorithms.

4.3.1. BSO with mitigation approaches (RP) and time-varying weights (TVW)

The effects of mitigation approaches and time-varying weights on the performance of the BSO algorithm are presented. In this investigation, all the test functions, except Schaffer's f6, were used in 40 dimensions, and maximum number of generations was set to 10,000. The average best fitness of the proposed algorithms for 100 trials for each test function is repre-

Table 2
Average fitness values, standard deviation, and the success rate of the algorithms for the benchmark functions for 100 trials.

Func.	Dim.	Max Iter.	BFA		ABC		BSO		BSO–RP		BSO–RPTVW	
			Avg.	Stdv.	Avg.	Stdv.	Avg.	Stdv.	Avg.	Stdv.	Avg.	Stdv.
f_{Sph}	10	5000	0.000031	0.00024	4.926E−28	2.187E−25	**8.475E−123**	3.953E−122	5.325E−116	8.422E−116	1.723E−118	7.401E−118
	20	7500	0.000892	0.00103	3.071E−25	5.368E−23	**2.361E−115**	7.348E−115	9.147E−114	4.473E−113	7.194E−114	1.510E−113
	30	10000	0.093786	0.02745	6.355E−24	1.240E−22	4.728E−102	1.372E−101	6.320E−107	2.928E−106	**3.218E−111**	6.672E−111
f_{Ros}	10	5000	7.2084513	9.436551	0.009252	0.010890	3.617E−7	5.081E−5	9.418E−8	2.192E−6	**1.858E−9**	3.204E−8
	20	7500	13.927110	16.830792	0.012870	0.017322	9.201E−7	1.102E−5	1.297E−8	8.053E−7	**8.952E−9**	1.460E−7
	30	10000	21.276334	25.03812	0.034010	0.048012	5.865E−6	8.519E−5	1.722E−7	6.940E−5	**6.027E−8**	4.392E−6
f_{Ras}	10	5000	0.003821	0.006513	4.398E−24	6.195E−23	4.171E−64	7.834E−64	**2.438E−95**	6.177E−95	8.5543E−93	4.814E−92
	20	7500	0.017613	0.100691	2.011E−22	7.048E−21	7.899E−61	2.215E−60	7.553E−88	3.510E−87	**6.903E−91**	3.149E−90
	30	10000	0.967820	4.24561	8.609E−22	5.332E−21	6.228E−59	8.496E−59	5.942E−83	1.837E−82	**2.631E−86**	7.258E−86
f_{Ack}	10	5000	0.000085	0.000237	8.462E−12	9.307E−12	7.105E−19	5.482E−18	9.021E−20	2.931E−18	**1.813E−20**	7.264E−19
	20	7500	0.000639	0.003406	3.285E−13	1.034E−12	2.131E−19	4.603E−18	5.286E−19	6.462E−18	**9.741E−20**	3.108E−19
	30	10000	0.001398	0.002054	7.632E−13	2.760E−12	1.460E−18	8.130E−17	2.372E−19	4.903E−17	**2.372E−19**	5.206E−17
f_{Gri}	10	5000	3.209850	4.298031	0.000219	0.000691	3.823E−46	6.679E−46	8.194E−47	2.468E−46	**7.991E−53**	3.931E−52
	20	7500	1.792011	1.973611	3.170E−9	1.725E−7	8.402E−47	3.928E−46	9.356E−49	4.514E−48	**6.812E−53**	4.740E−52
	30	10000	0.981425	1.023101	5.578E−10	3.008E−8	4.161E−47	7.523E−47	6.322E−49	1.097E−48	**5.695E−54**	2.243E−53
f_{Shf6}	2	2000	9.381E−8	1.0348E−7	3.806E−16	3.183E−14	1.069E−49	3.170E−49	5.398E−51	7.286E−51	**3.546E−56**	6.538E−56

Table 3
The success rate and the average number of iteration before the algorithms reach the success criteria for the benchmark functions for total of 100 trials.

Func.	Dim.	Max Iter.	BFA		ABC		BSO		BSO–RP		BSO–RPTVW	
			Avg. Iter.	Succ.Rate	Avg.Iter.	Succ. Rate	Avg. Iter	Succ. Rate	Avg. Iter	Succ. Rate	Avg. Iter	Succ. Rate
f_{Sph}	10	5000	196.9	1	124.6	1	**95.7**	1	96.2	1	98.1	1
	20	7500	423.8	0.73	387.0	1	**313.2**	1	317.8	1	321.4	1
	30	10000	783.4	0.21	556.2	1	**490.6**	1	498.3	1	512.6	1
f_{Ros}	10	5000	X	0.00	X	0.00	549.3	1	193.6	1	**91.0**	1
	20	7500	X	0.00	X	0.00	804.5	1	365.3	1	**103.2**	1
	30	10000	X	0.00	X	0.00	1225.2	1	548.0	1	**123.8**	1
f_{Ras}	10	5000	3693.4	0.07	498.6	1	**487.1**	1	567.3	1	605.7	1
	20	7500	3782.5	0.03	1109.3	1	**946.4**	1	973.9	1	1038.2	1
	30	10000	X	0.00	1590.4	1	**1243.8**	1	1270.5	1	1314.6	1
f_{Ack}	10	5000	2725.8	0.35	461.2	1	207.6	1	**199.1**	1	206.5	1
	20	7500	3137.4	0.14	703.5	1	343.2	1	329.3	1	**323.2**	1
	30	10000	3485.2	0.08	942.6	1	438.9	1	438.1	1	**439.8**	1
f_{Gri}	10	5000	X	0.00	478.0	0.47	431.8	1	220.3	1	**215.0**	1
	20	7500	X	0.00	1031.3	1	515.3	1	350.1	1	**348.1**	1
	30	10000	X	0.00	1485.8	1	611.4	1	445.7	1	**443.9**	1
f_{Shf6}	2	2000	198.3	1	126.7	1	93.0	1	86.3	1	**85.4**	1

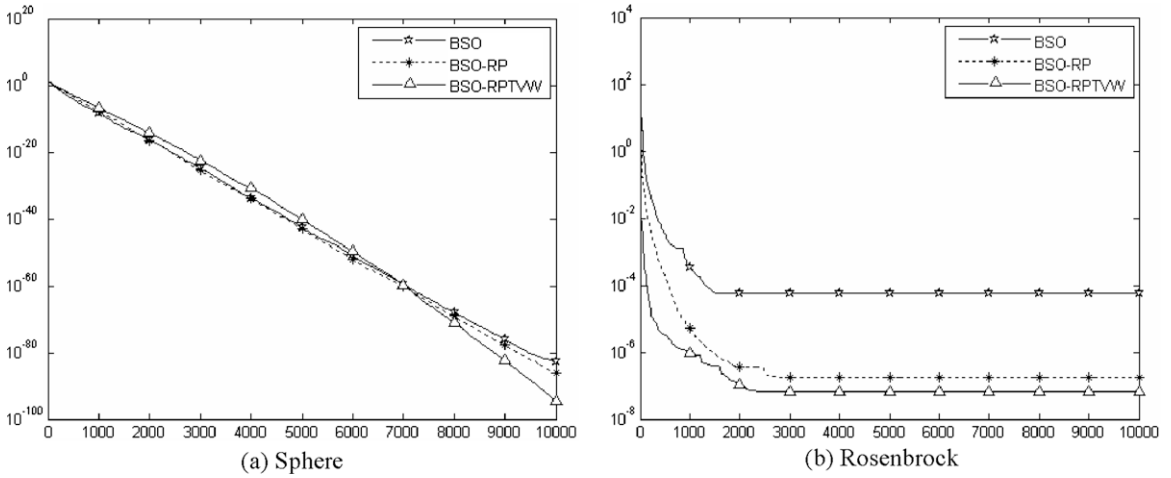

Fig. 4. The logarithm of average fitness for sphere and Rosenbrock functions. Horizontal and vertical axes, respectively represent number of iterations and logarithm of the average fitness.

sented in Figs. 4–6. The vertical axis in each diagram represents the logarithm of average fitness, and the horizontal axis represents the iteration number.

Figs. 4–6 show that BSO–RP and BSO–RPTVW converge faster then BSO algorithm at the early stage of optimization process for Rosenbrock and Griewank functions. For the Sphere function, BSO and BSO–RP almost converge faster than BSO–RPTVW. For Ackley and Schaffer's f6, all the algorithms have almost same convergence speed, and finally, for Rastrigrin function, faster convergence is provided by BSO and BSO–RPTVW. As a consequence, it is clear from Figs. 4–6 that BSO and its two extensions have fast convergence rate, and significant improvement in convergence speed is not achieved by adding mitigation approaches and time-varying weights. However, BSO–RPTVW is slightly faster than BSO and BSO–RP.

It is clear from Figs. 4–6, that the average optimum solution in terms of the predefined number of generations is improved by applying mitigation approaches and time-varying weights. The BSO–RP and BSO–RPTVW produce better results then BSO for all the test functions. The BSO–RP and BSO–RPTVW produce the same result for Ackley function. The percentage of improvement provided by two extensions was 10% for Ackley function. The BSO–RP improves the performance of BSO more than 10% in optimizing Rosenbrock function, further improvement is achieved by time-varying weights. Up to 4%, 8%, 15% improvements respectively were obtained by BSO–RP for optimizing Sphere, Griewank, and Rastrigrin functions. Further improvements were obtained by BSO–RPTVW algorithm (10%, 20%, 28% for optimizing Sphere, Griewank, and Rastrigrin functions). For the Schaffer's f6 function BSO and BSO–RP relatively produce same results, the improvement up to 10% was obtained using time-varying weights.

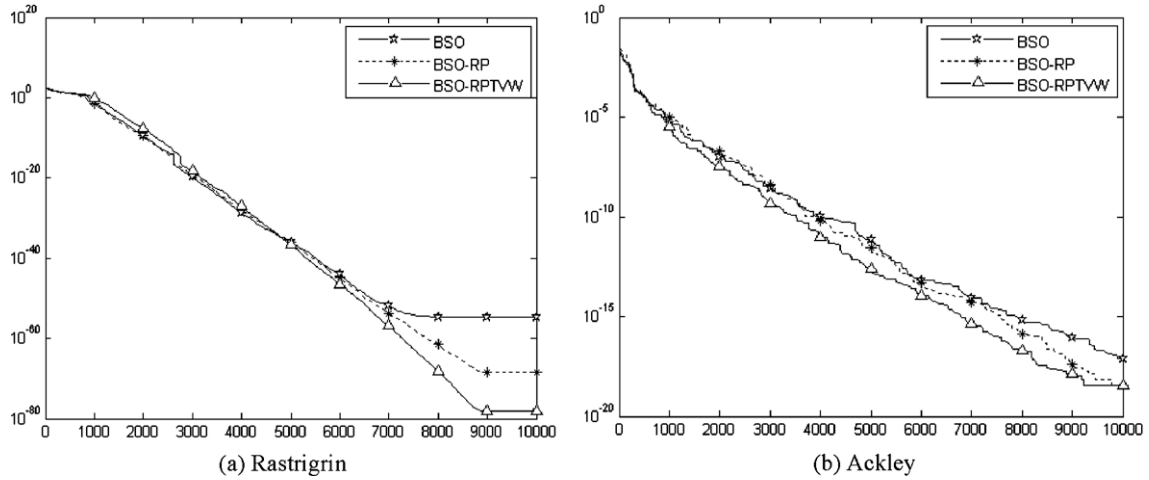

Fig. 5. The logarithm of average fitness for Rastrigrin and Ackley functions. Horizontal and vertical axes, respectively represent number of iterations and logarithm of the average fitness.

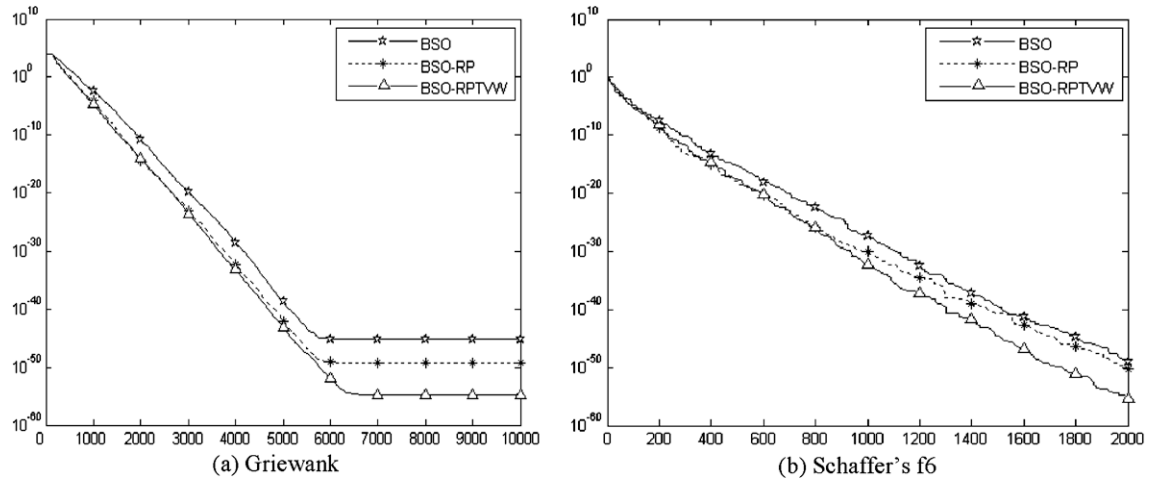

Fig. 6. The logarithm of average fitness for Griewank and Schaffer's f6 functions. Horizontal and vertical axes, respectively represent number of iterations and logarithm of the average fitness.

In general, the results show that BSO optimizer and its two extensions act as highly consistent strategies in terms of optimum solution, and provide robust behavior which is not strictly dependent on the type of the test function. Using mitigation approaches and time-varying weights, excellent results were observed for all test functions considered in this investigation. As a consequence, the BSO–RPTVW algorithm produces better results than BSO, BSO–RP, and other algorithms investigated in this consideration.

5. 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. Also two further extensions were suggested which aim to improve the performance of the BSO algorithm in terms of minimizing numerical functions within a predefined number of iterations. 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.

Stagnation and premature convergence are the main deficiencies of the population based optimization algorithms when solving numerical functions. To alleviate the stagnation, we proposed two new strategies which improve the performance of BSO. First, the concept of penalty was introduced which encourages a stagnant bee to leave its position and move to new

positions with better nectar. Second, another concept "repulsion factor" was introduced as a strategy to mitigate stagnation. These extensions introduce significant improvement of performance compared with BSO. To mitigate the premature convergence, linear weights were introduced to the BSO. Using dynamic weights provides good improvement in terms of optimum solution for all the test functions.

The BSO algorithm and its extensions were compared with two other bee colony optimization algorithms on the six benchmark functions. The experimental results showed that the introduction of mitigation strategies into the BSO algorithm improves the performance in terms of optimum solution. Also the experiments results showed that the BSO and its extensions are effective and powerful algorithm for numerical function optimization and outperform other algorithms investigated in this study.

References

[1] Srinivasan D, Seow TH. Evolutionary Computation, CEC '03, 8–12 Dec. 2003, Canberra, Australia; 2003. p. 2292–7.
[2] Boyer DO, Martnez CH, Pedrajas NG, Crossover operator for evolutionary algorithms based on population features. Available from: <http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume24/ortizboyer05a-html/Ortiz-Boyer.html>.
[3] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceeding of IEEE international conference neural networks, vol. 4; 1995. p. 1942–7.
[4] Eberhart RC, Shi Y. Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of IEEE congress on evolutionary computation, Seoul, Korea; 2001. p. 94–7.
[5] Ratnaweera A, Halgamuge SK, Watson HC. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. In: IEEE transactions on evolutionary computation, vol. 8, no. 3; 2004. p. 240–55.
[6] Suganthan PN. Particle swarm optimizer with neighborhood operator. In: Proceedings of the IEEE international congress evolutionary computation, vol. 3; 1999. p. 1958–62.
[7] Suganthan PN. Particle swarm optimizer with neighborhood operator. In: Proceedings of the IEEE international congress evolutionary computation, vol. 3; 1999. p. 1958–62.
[8] Cox Melissa D, Myerscough Mary R. A flexible model of foraging by a honey bee colony: the effects of individual behaviour on foraging success. J Theor Biol 2003;223:179–97.
[9] Karaboga N. A new design method based on artificial bee colony algorithm for digital IIR filters. J Franklin Instit 2009;346:328–48.
[10] Baykasoglu A, Ozbakır L, Tapkan P. Swarm intelligence: focus on ant and particle swarm optimization; 2007.
[11] Teodorovic D, Dell Orco M. Bee colony optimization – a cooperative learning approach to complex transportation problems. J Adv OR AI Meth Transport 2007:51–60.
[12] Teodorovic Dusan, Lucic Panta, Markovic Goran, Orco Mauro Dell'. Bee colony optimization: principles and applications. In: Proceeding of eighth seminar on neural network applications in electrical engineering, Neurel; 2006. p. 151–6.
[13] Pham DT, Castellani M, Fahmy AA. Learning the inverse kinematics of a Robot manipulator using the bees algorithm. In: The IEEE international conference on industrial informatics; 2008. p. 493–8.
[14] Bahamish Hesham Awadh A, Abdullah Rosni, Abdul Salam Rosalina, Protein conformational search using bees algorithm. In: Proceeding of second Asia international conference on modelling & simulation; 2005. p. 238–44.
[15] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71.
[16] Yang XS. Engineering optimizations via nature-inspired virtual bee algorithms. In: Lecture notes in computer science, Springer (GmbH); 2005. p. 317–23.
[17] Sundareswaran K, Sreedevi VT. Development of novel optimization procedure based on honey bee foraging behavior. In: IEEE International conference on systems, man and cybernetics; 2008. p. 1220–5.
[18] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. J Appl Mathe Comput 2009.
[19] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. J Appl Soft Comput 2008;8:687–97.
[20] Ho1 SL, Yang S, Ni G, Lo EWC, Wong HC. A particle swarm optimization-based method for multiobjective design optimizations. IEEE Trans Magnet 2005;41(5):1756–9.
[21] Sun TY, Lin CL, Liu CC. Effective learning rate adjustment of blind source separation based on an improved particle swarm optimizer. IEEE Trans Evolut Comput 2006.
[22] Riget J, Vesterstrom JS. A diversity-guided particle swarm optimizer – the ARPSO EVALife. Tech. Rep. 2002-02.
[23] Eberhart Russell C, Shi Yuhui. Guest editorial special issue on particle swarm optimization. IEEE Trans Evolut Comput 2004;8(3):201–3.
[24] Parsaopoulos KE, Vrahatis MN. Recent approaches to global optimization problems through particle swarm optimization. J Nat Comput 2002:235–306.
[25] Eberhart RC, Shi Y. Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of IEEE congress on evolutionary computation, Seoul, Korea; 2001. p. 94–7.
[26] Akbari R, Ziarati K. Combination of particle swarm optimization and stochastic local search for multimodal function optimization. In: Proceeding of IEEE Pacific-Asia workshop on computational intelligence and industrial application; 2008. p. 388–92.