UDC 57

Li-Chuan L., Unurjargal D. On the Performance Evaluation of a Collaborative Swarm Intelligence Approach Particle Bee Algorithm

Li-Chuan Lien,

[1] Assistant Professor, Dep. of Civil engineering, Chung Yuan Christian University, Taoyuan, Taiwan

Unurjargal Dolgorsuren

[2] Ph.D student, Dep. of Civil engineering, Chung Yuan Christian University, Taoyuan, Taiwan

Abstract. *Swarm intelligence (SI), an artificial intelligence (AI) approach widely used in many complex optimization problems, models the collective behavior of social systems such as honeybees and birds. This study evaluated a collaborative swarm intelligence approach optimization algorithm, named the particle bee algorithm (PBA). The PBA is based on a particular aspect of bird (particle swarm optimization, PSO) and honeybee swarm (bee algorithm, BA) behaviors that integrates their advantages and proposes a self-parameter-updating technique to prevent being trapped into a local optimum in high dimensional problems. This study compares the performance of PBA with that of differential evolution (DE), evolutionary algorithms (EA), particle swarm optimization (PSO) and bee algorithm (BA) for multi-dimensional numeric problems. For test problems carried out in this work, colony sizes ranging from 75 to 100 of PBA can provide an acceptable convergence speed for an optimization search. Besides, elite and best bee PSO iteration sizes of (15, 9) to (30, 18) can provide an acceptable convergence speed for an optimization search. Results show PBA performance to be comparable to that of mentioned algorithms, and the potential for its being efficiently employed to solve benchmark numerical problems with high dimensionality.*

Keywords: *Swarm intelligence, Bee algorithm, Particle swarm optimization, Particle bee algorithm.*

DOI 10.54092/25421085_2021_11_49

Рецензент: Ерофеевская Лариса Анатольевна - кандидат биологических наук, старший научный сотрудник. Федеральное государственное бюджетное учреждение науки Федеральный исследовательский центр «Якутский научный центр Сибирского отделения Российской академии наук», Институт проблем нефти и газа Сибирского отделения Российской академии наук (ИПНГ СО РАН)

1. INTRODUCTION

Evolutionary algorithms (EAs), generally known as general-purpose optimization algorithms, are often used to find, within a reasonable compilation time, near-optimal solutions to numerical, real-valued test problems. Differential evolution algorithms (DEs) are one type of recently introduced EA (Price, etc. 2005). DEs have been proposed to overcome the poor local search ability of genetic algorithms (GAs) (Holland, 1975). Selection operations used represent an important difference between GAs and DEs. For GAs, the chance of being selected as a parent solution depends on the relevant solution's fitness value (Krink, etc., 2004). In DEs, all solutions have an equal chance of being selected as parents, i.e., the chance does not depend on fitness values. After a new solution is produced using self-adjusting mutation and crossover operations, the new solution competes with its parent for the next generation, with the better one winning the competition. In other words, a greedy scheme is applied to select one of the two for the next generation. Using a mutation operation, which is able to self adapt, perform crossover operations and make selections via a greedy process, makes DEs fast-converging evolutionary algorithms (Krink, etc., 2004). This has made them the subject of significant interest by researchers from a diverse range of fields, who have applied DEs to a variety of real world problems (Price, etc., 2005; Krink, etc., 2004).

Swarm intelligence (SI) has been of increasing interest to research scientists in recent years. Swarm intelligence was defined by Bonabeau et al. as any attempt to design algorithms or distributed problem-solving devices based on the collective behavior of social insect colonies or other animals (Bonabeau, etc., 2004). Bonabeau et al. focused primarily on the social behavior of ants (Dorigo, 1992), fish (Li, 2003), birds (Kennedy, etc., 1995) and bees (Pham, etc., 2006) etc. However, the term “swarm” can be applied more generally to refer to any restrained collection of interacting agents or individuals. Although bees swarming around a hive is the classical example of “swarm”, swarms can easily be extended to other systems with similar architectures.

A few models have been developed to model the intelligent behaviors of honeybee swarms and applied to solve combinatorial type problems. Yang (2006) presented a virtual bee algorithm (VBA) that is effective when applied to function optimization problems. However, while the proposed algorithm was similar to GA, it was much more efficient due to the parallelism of multiple independent bees. VBA was tested on two functions with two parameters, single-peaked and multi-peaked, respectively. Results show the VBA as significantly more efficient than GA. Karaboga et al. (2009) presented an artificial bee colony (ABC) algorithm and expanded its experimental results (Basturk, etc., 2006). It has been pointed out that the ABC algorithm outperforms GA for functions exhibiting multi-modality or uni-modality. Pham et al. (2006) presented an original bee algorithm (BA) and applied to two standard functional optimization problems with two and six dimensions. Results demonstrated the BA able to find solutions very close to the optimum, showing that BA generally outperformed GA. Ozbakir et al. (2010) developed a modified BA (Pham, etc., 2006) to solve generalized assignment problems (GAP) that presented an ejection chain neighborhood mechanism. This study found that the proposed BA offers the potential to solve GAP. However, while BA (Pham, etc., 2006) offers the potential to conduct global searches and uses a simpler mechanism in comparison with GA, it is weak in local searching and does not records past searching experiences during the optimization search process.

For instance, a flock of birds may be thought of as a swarm whose individual agents are birds. Particle swarm optimization (PSO), which has become quite popular for many researchers recently (Tsai, 2010; Parsopoulos, etc. 2007), models the social behavior of birds (Kennedy, 1995). PSO is a population-based stochastic optimization technique that is well adapted to the optimization of nonlinear functions in multi-dimensional space. PSO consists of a swarm of particles moving in a search space of potential problem solutions. Every particle has a position vector representing a candidate solution to the problem and a velocity vector. Moreover, each particle contains a small memory that stores its own best position so far and a global best position obtained through communication with neighbor particles. PSO potentially used in local searching, and records past searching experiences during optimization search process. However, it converges early in highly discrete problems (Korenaga, etc., 2006).

Hence, in order to improve BA and PSO, Cheng (2012) and Lien (2012, 2014) proposed an optimization hybrid swarm algorithm, named the particle bee algorithm (PBA), based on intelligent behavior traits of bird and honeybee swarms. PBA has been successful applied to many case studies (Cheng and Lien, 2012; Lien and Cheng, 2012, 2014). PBA integrates their advantages and a self-parameter-updating technique to prevent becoming trapped in a local optimum in high dimensional problems. This study compares the performance of the PBA algorithm with that of DE, EA, PSO (Krink, etc., 2004) and BA (Pham, etc., 2006) for a set of well-known test functions (Krink, etc., 2004). Also, the performance of PBA is analyzed under conditions in which control parameter values change. In Section 2 and 3, bee algorithm (BA) and particle swarm optimization (PSO) are described and then the particle bee algorithm (PBA) is introduced in Section 4. In Section 5, the experimental study is described. Obtained simulation results are presented and discussed in Section 6.

2. BEE ALGORITHM (BA)

Bee algorithm (BA) is an optimization algorithm inspired by the natural foraging behavior of honeybees as they work to find an optimal solution (Eberhart, 2006). The BA flowchart shows in Fig. 1. The BA (Pham, etc., 2006) requires the setting of a number of parameters, including number of scout bees (*n*), number of elite sites selected from n visited sites (*e*), number of best sites out of n visited sites (*b*), number of bees recruited for elite e sites (n_1), number of bees recruited for best b sites (n_2), number of bees recruited for other visited sites (*r*), and neighborhood (*ngh*) of bees dance search and stopping criterion.

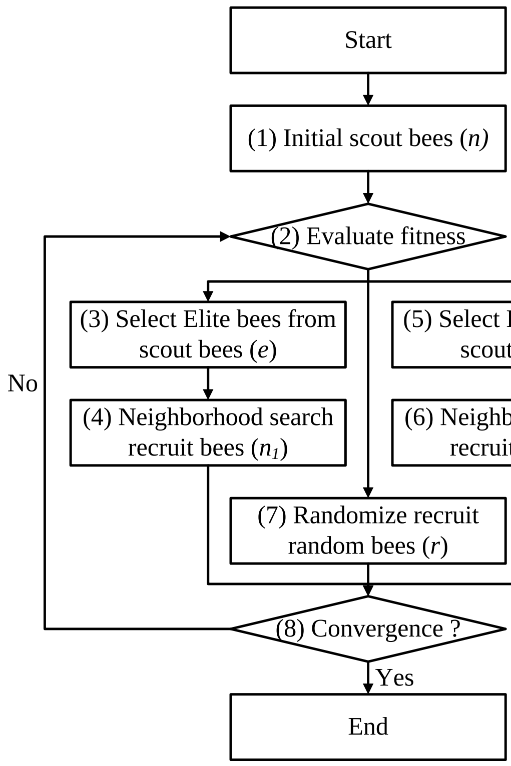

Start
(1) Initial scout bees (n)
(2) Evaluate fitness
(3) Select Elite bees from scout bees (e)
(5) Select Best bees from scout bees (b)
No
(4) Neighborhood search recruit bees (n1)
(6) Neighborhood search recruit bees (n2)
(7) Randomize recruit random bees (r)
(8) Convergence ?
Yes
End

Fig.1. Bee algorithm flowchart

Step (1) Initialize scout bees

The BA starts with *n* scout bees placed randomly in the search space.

Step (2) Evaluate fitness

Start the loop and evaluate scout bee fitness.

Step (3) Select elite sites (*e*) from scout bees

Bees that have the highest fitness are chosen as elite bees, and sites they visit are chosen for neighborhood search.

Step (4) Recruit bees (n_1) begin neighborhood dancing search

The algorithm conducts searches in the neighborhood of selected sites, assigning more recruit bees to dance near to elite sites. Recruit bees can be chosen directly according to the fitness associated with dancing sites Eq. (1).

$$x_{id}(t+1) = x_{id}(t) \times (Rand - 0.5) \times 2 + x_{id}(t) \quad \text{(1)}$$

where x_i is ith x and i = 1 to n; d is dimension in x_i and d = 1 to D, t is iteration; $x_{id}(t+1)$ is dth dimension in ith x and in t+1 iteration; $x_{id}(t)$ is dth dimension in ith x and in t iteration; *Rand* is a uniformly distributed real random number within the range 0 to 1; n is number of scout bees.

Step (5) Select best sites (*b*) from scout bees

Otherwise, elite bees with the highest fitness are chosen as best bees, and sites they visit are chosen for neighborhood search.

Step (6) Recruit bees (n_2) begin neighborhood dancing search

The algorithm conducts searches in the neighborhood of the selected sites, assigning more recruit bees to dance near the best sites. Recruit bees can be chosen directly according to the fitness associated with dancing sites Eq. (1).

Elite bees differ from best bees as the former focus on local search in order to search the local optimum solution, and the latter focus on global search in order to avoid missing other potential global optimum solutions. Alternatively, fitness values are used to determine the elite/best bees selected. Dancing searches in the neighborhood of elite and best sites that represent more promising solutions are made more detailed by recruiting more bees to follow them than others.

Step (7) Recruit random bees (r) for other visited sites

The remaining bees in the population are assigned randomly around the search space scouting for new potential solutions.

Step (8) Convergence?

Throughout step (3) to step (7), such differential recruitment is a key BA operation. However, in step (8) only bees with the highest fitness for the current iteration will be selected for the following iteration. While there is no such restriction in nature, it is introduced here to reduce the number of points to be explored. These steps are repeated until a stopping criterion is met that determines whether bees are to be abandoned or memorized.

From Eq. (1), BA dependence on random search makes it relatively weak in local search activities. Also, BA does not have past searching records of PSO capabilities.

3. PARTICLE SWARM OPTIMIZATION (PSO)

Particle swarm optimization (PSO) is an optimization algorithm inspired by the natural foraging behavior of birds to find an optimal solution (Kennedy, 1995). In PSO, a population of particles starts to move in a search space by following current optimum particles and changing their positions in order to find the optimum. The position of a particle refers to a possible solution of the function to be optimized. Evaluating the function by the particle's position provides the fitness of that solution. In every iteration, each particle is updated by following the best current particle solution achieved so far (local best) and the best of the population (global best). When a particle takes part of the population as its topological neighbors, the best value becomes a local best. Particles tend to move toward good areas in the search space in response to information spreading through the swarm. A particle moves to a new position calculated by the velocity updated at each time step *t* by Eq. (2). Eq. (3) is then used to calculate the new velocity, as the sum of the previous position and the velocity.

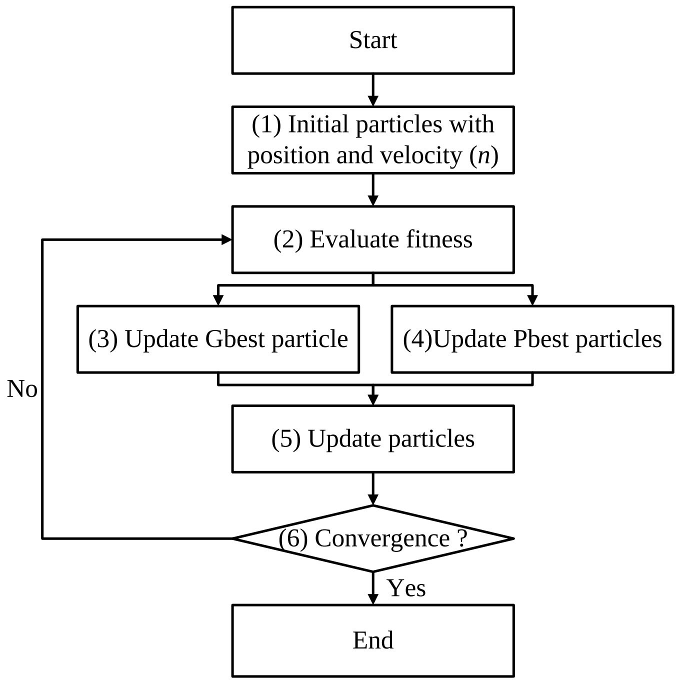
$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \ldots\ldots (2)$$

where x_i is ith x and $i = 1$ to n; v_i is ith v; d is dimension in x_i or v and $d = 1$ to D; t is iteration; $x_{id}(t)$ is dth dimension in ith x and in t iteration; v_{id}(t+1) is dth dimension in ith v and in t+1 iteration; $x_{id}(t+1)$ is dth dimension in ith x and in t+1 iteration; n is number of particles.

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times Rand \times [P_{id}(t) - x_{id}(t)] + c_2 \times Rand \times [G_d(t) - x_{id}(t)] \ldots\ldots (3)$$

where v_{id}(t) is dth dimension in ith v and in t iteration; w is inertia weight and controls the magnitude of the old velocity $v_{id}(t)$ in the calculation of the new velocity; $P_{id}(t)$is dth dimension in ith local best particle and in t iteration; $G_d(t)$ is dth dimension global best particle in t iteration; c_1 and c_2 determine the significance of P_{id}(t) and G_d(t); *Rand* is a uniformly distributed real random number within the range 0 to 1.

Furthermore, v_{id} at any time-step of the algorithm is constrained by parameters v_{max} and v_{min}. The swarm in PSO is initialized by assigning each particle to a uniformly and randomly chosen position in the search space. Velocities are initialized randomly in the range v_{max} to v_{min}. Particle velocities on each dimension are clamped to a maximum velocity v_{max}. If the velocity of that dimension exceeds v_{max} or v_{min} (user-specified parameters), dimension velocity is limited to v_{max} or v_{min}. Fig. 2 shows the PSO flowchart.


Start
(1) Initial particles with position and velocity (n)
(2) Evaluate fitness
(3) Update Gbest particle
(4)Update Pbest particles
(5) Update particles
(6) Convergence ?
No
Yes
End
Fig.2. Particle swarm optimization flowchart

Step (1) Initialize particles

The PSO starts with *n* particles being randomly introduced with respective positions and velocities into the search space.

Step (2) Evaluate fitness

Start the loop and evaluate particle fitness.

Step (3) Update Gbest particle

The algorithm updates global best particle through problem iterations.

Step (4) Update Pbest particles

The algorithm updates local best particles through the current problem iteration.

Step (5) Update particles using steps (3) and (4)

The algorithm updates particles using Eq. (2) and Eq. (3).

Step (6) Convergence?

The above steps are repeated until the stop criterion is met.

However, while PSO may be employed in local search and has a track record of experience being used in optimization search processes, it tends to achieve early convergence in highly discrete problems (Korenaga, 2006).

4. PROPOSED PARTICLE BEE ALGORITHM (PBA)

In order to integrate BA global search ability with the local search advantages of PSO, Cheng (2012) and Lien (2012, 2014) proposed an optimization hybrid swarm algorithm, the particle bee algorithm (PBA), based on the intelligent behaviors of bird and honeybee swarms. For improved BA local search ability, PSO global search ability and to seek records past experience during optimization search process, this study reconfigures the neighborhood dance search (Pham, etc., 2006) as a PSO search (Kennedy, 1995). Based on cooperation between bees (BA) and birds (PSO), the proposed algorithm improves BA neighborhood search using PSO search. Therefore, PBA employs no recruit bee searching around “elite” or ”best” positions (as BA does). Instead, a PSO search is used for all elite and best bees. In other words, after PSO search, the number of “elite”, “best” and “random” bees equals the number of scout bees.

In PBA, the particle bee colony contains four groups, namely (1) number of scout bees (*n*), (2) number of elite sites selected out of n visited sites (*e*), (3) number of best sites out of n visited sites (*b*), and (4) number of bees recruited for the other visited sites (*r*). The first half of the bee colony consists of elite bees, and the second half includes the best and random bees. The particle bee colony contains two parameters, i.e., number of iteration for elite bees by PSO (*Peitr*) and number of iteration for best bees by PSO (*Pbitr*). Fig. 3 shows the PBA flowchart.

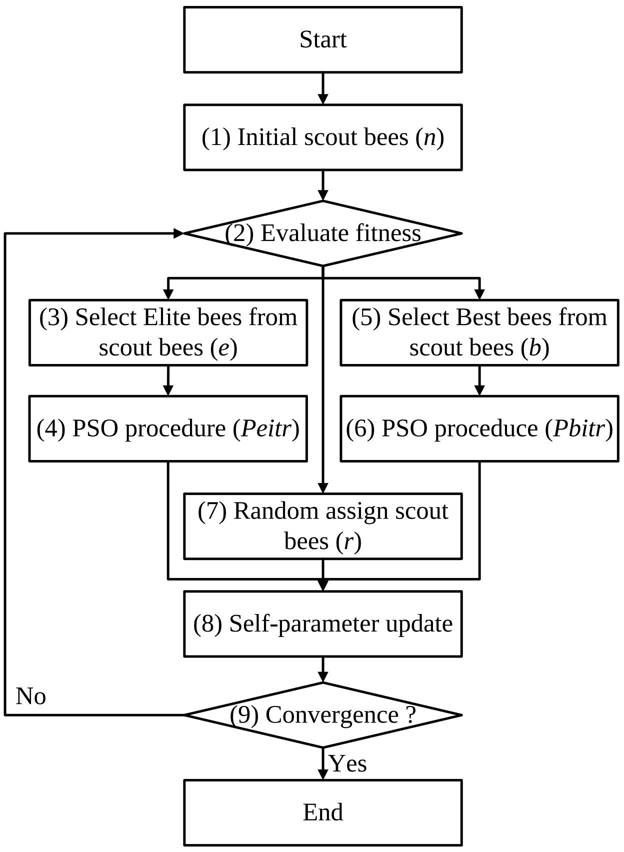

Start
(1) Initial scout bees (n)
(2) Evaluate fitness
(3) Select Elite bees from scout bees (e)
(5) Select Best bees from scout bees (b)
(4) PSO procedure (Peitr)
(6) PSO proceduce (Pbitr)
(7) Random assign scout bees (r)
(8) Self-parameter update
No
(9) Convergence ?
Yes
End

Fig.3. Particle Bee algorithm flowchart

Step (1) Initialize scout bees

The PBA starts with *n* scout bees being randomly placed with respective positions and velocities in the search space.

Step (2) Evaluate fitness

Start the loop and evaluate scout bee fitness.

Step (3) Select elite sites (*e*) from scout bees.

Elite sites are selected for each elite bee, whose total number is equal to half the number of scout bees.

Step (4) Elite bees begin the PSO procedure using *Peitr* iteration.

In this step, Eq. (2) is used to produce new particle bees from elite and best bees. Elite and best bees velocity updates are performed as indicated in Eq. (3).

Step (5) Select best sites (*b*) from scout bees.

Best sites are selected for each best bee, the total number of which equals one-quarter of the number of scout bees.

Step (6) Best bees begin the PSO procedure using *Pbitr* iteration.

In this step, new particle bees from elite and best bees are produced using Eq. (2). Elite and best bee velocities are updated as indicated in Eq. (3).

Step (7) Recruit random bees (*r*) for other visited sites

The random bees in the population are assigned randomly around the search space scouting for new potential solutions. The total number of random bees is one-quarter of the number of scout bees.

Step (8) Self-parameter-updating for elite, best and random bees

Furthermore, in order to prevent trapping into a local optimum in high dimensional problems, this study proposed a solution, namely a self-parameter updating technique, the idea for which comes from Ref. (Karaboga, etc., 2009). Eq. (4) describes the self-parameter updating equation.

$x_{id(new)} = x_{id(cur)} + 2\times(Rand-0.5)\times(x_{id(old)} - x_{jk})$ .... (4)

$j = \text{int}\ (Rand \times n) + 1$ ....(5)

$k = \text{int}\ (Rand \times d) + 1$ ....(6)

where x_i is i_{th} x and $i = 1$ to n; d is dimension in x_i and $d = 1$ to D; $x_{id}(cur)$ is d_{th} dimension in i_{th} x and in current solution; $x_{id}(new)$ is d_{th} dimension in i_{th} x and in new solution; *Rand* is a uniformly distributed real random number within the range 0 to 1; j is the index of the solution chosen randomly from the colony as shows in Eq. (5), k is the index of the dimension chosen randomly from the dimension as shows in Eq. (6); n is number of scout bees.

In step (8), after elite, best and random bees are distributed according to fineness, finesses are checked to indicate whether they should be abandoned or memorized using Eq. (4). If finesses of elite, best or random bees are improved by Eq. (4) to a degree that is superior to previous finesses, then these current finesses will be memorized. This differential recruitment is a key operation of the PBA between steps (3) and (8).

Step (9) Convergence?

In this step, only the bee with the highest fitness will be selected to form the next bee population. These steps are repeated until the stop criterion is met and bees are selected to be abandoned or memorized.

In PBA, scout bees are used to classify both elite and best bees. Classification is controlled by scout bee fitness and optimized by control parameters called ‘‘*Peitr*” and “*Pbitr*”, which are important PBA control parameters. In the PBA, the idea of *Peitr* for elite bees gives a higher potential to search optimization solutions. The idea of *Pbitr* for best bees gives a second opportunity to search optimization solutions because luck continues to play a role in resource identification. Therefore, in this study, *Peitr* is always larger then *Pbitr*. In a robust search process, exploration and exploitation processes must be carried out together. In the PBA, while elite bees (*Peitr*) implement the exploitation process in the search space, best bees (*Pbitr*) and random bees control this process.

5. EXPERIMENTS

In order to evaluate PBA performance, some classical benchmark functions given by Krink (2004) are presented in Table 1. PBA results compare favorably with the DE and EA results of Krink (2004). This study equalized the total number of evaluations to 100,000 for the first two functions and 500,000 for the other three functions, as shown in Ref. (Krink, etc., 2004). In the PSO and BA, cycles were designated as 1,000 for f_1 and f_2 and 5,000 for f_3, f_4 and f_5. In PBA, maximum number of cycles was designated for *Bitr*, *Peitr*, *Pbitr* as, respectively, 1,000, 15, and 9 for f_1 and f_2 and 5,000, 15, 9 for f_3, f_4 and f_5. The number of elite bees totaled 50 percent of the colony and the number of best bees totaled 25 percent of the colony. In PBA, the number of random bees is taken n-e-b from the number of elite bees and best bees. The PBA used in the simulation studies and values assigned for the parameter settings of DE, EA, PSO in Ref. (Krink, etc., 2004) and BA in the Ref. (Pham, etc., 2006) are given in Table 2. From the table, it can see that assigned values for DE, EA and PSO in Ref. (Krink, etc., 2004) and BA in Ref. (Pham, etc., 2006) are the same as recommended values in the literature for the associated control parameters.

In experiments, f_1 is a 2 dimensional Schaffer function, f_2 is a 5 dimensional

Sphere function, f3(x), f4(x) and f5(x) are 50 dimensional Griewank, Rastrigin and Rosenbrock functions. Parameter ranges, formulations and global optimum values of these functions are given in Table 1. Schaffer and Sphere functions usually test for search performance as basic optimization problems. Since local optima numbers increase with dimensionality, the Griewant function is strongly multimodal. The multimodality disappears for sufficiently high dimensionalities (D>30) and the problem seems unimodal. The Rastrigin function is based on the Sphere function with the addition of cosine modulation to produce many local minima. Thus, the function is multimodal. Minima locations are regularly distributed. The difficult part about finding optimal solutions to this function is that an optimization algorithm can easily become trapped in a local optimum on its way toward the global optimum. The Rosenbrock function is well-known classic optimization problem. The global optimum is inside a long, narrow, parabolic shaped flat valley. Due to the difficulties in converging on the global optimum of this function, variables are strongly dependent, and gradients generally do not point toward the optimum. Schaffer, Sphere, Griewank, Rastrigin and Rosenbrock functions, surface plot, and contour line are shown in Fig.4 to Fig.8.

Table1

Numerical benchmark functions

Function	Formula	Range	Minimum	Dimension
Schaffer	$f_1(\vec{x}) = 0.5 + \frac{\sin^2(\sqrt{x_{i1}^2 + x_{i2}^2}) - 0.5}{(1 + 0.001(x_{i1}^2 + x_{i2}^2))^2}$	[-100, 100]	$f_1(\vec{0}) = 0$	2
Sphere	$f_2(\vec{x}) = \sum_{d=1}^{5} x_{id}^2$	[-100, 100]	$f_2(\vec{0}) = 0$	5
Griewank	$f_3(\vec{x}) = \frac{1}{4000}\left(\sum_{d=1}^{50}(x_{id} - 100)^2\right) - \left(\prod_{d=1}^{50}\cos(\frac{x_{id} - 100}{\sqrt{i}})\right) + 1$	[-600, 600]	$f_3(\overrightarrow{100}) = 0$	50
Rastrigin	$f_4(\vec{x}) = \sum_{d=1}^{50}(x_{id}^2 - 10\cos(2\pi x_{id}) + 10)$	[-5.12 , 5.12]	$f_4(\vec{0}) = 0$	50
Rosenbrock	$f_5(\vec{x}) = \sum_{d=1}^{50} 100(x_{i(d+1)} - x_{id}^2)^2 + (x_{id} - 1)^2$	[-50, 50]	$f_5(\vec{1}) = 0$	50

Note: where i = 1 to n.

Table 2

Parameter values used in the experiments

DE		EA		PSO		BA		PBA	
n	50	*n*	100	*n*	20	*n*	100	*n*	100
CF	0.8	*Pc*	1.0	*w*	1.0~0.7	*e*	n/2	*e*	n/2
f	0.5	*Pm*	0.3	*v*	X_{min}/10~X_{max}/10	*b*	n/4	*b*	n/4
		M	0.01			*r*	n/4	*r*	n/4
		N	10			n_1	2	*w*	1.0~0.7
						n_2	1	*v*	X_{min}/10~X_{max}/10
								Peitr	15
								Pbitr	9

n=population size (colony size); *CF*=crossover factor for DE; *pc*=crossover rate for EA; *pm*=mutation rate; *M*=mutation variance; *N*=elite size; *f*=scaling factor; *w*=inertia weight; *v*=limit of velocity for PSO; *e*=elite bee number; *b*=best bee number; *r*=random bee number; n_1= elite bee neighborhood number; n_2=best bee neighborhood number; *Peitr*=PSO iteration of elite bees; *Pbitr*=PSO iteration of best bees.

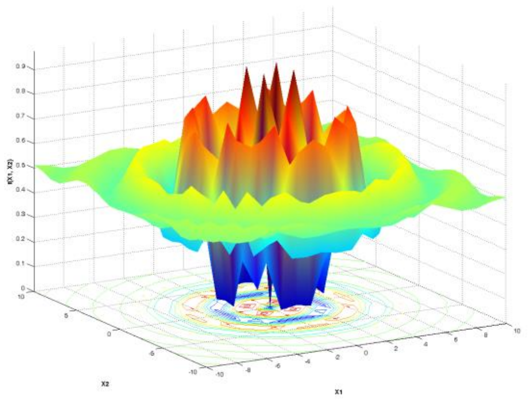


Fig.4. Schaffer function

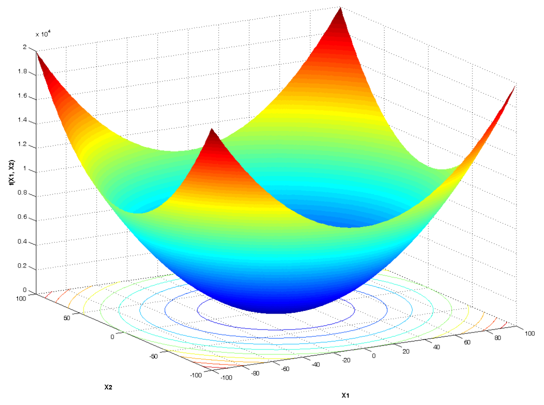


Fig.5. Sphere function

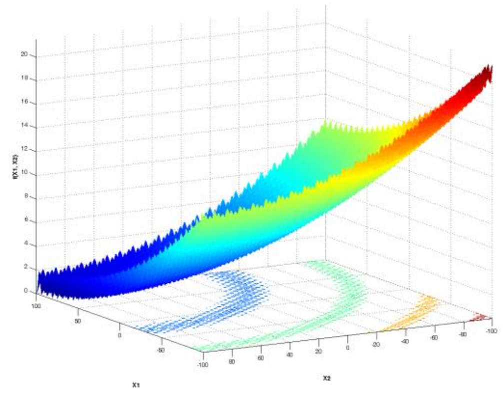


Fig.6. Griewank function

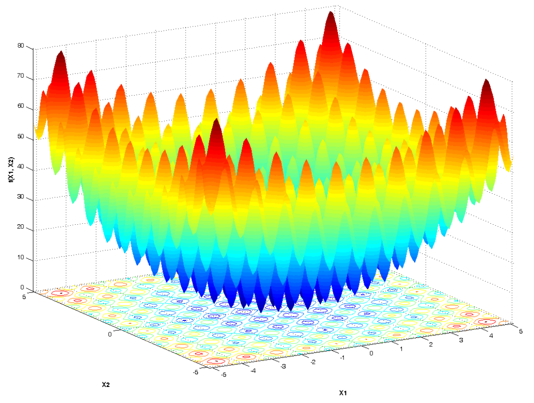


Fig.7. Rastrigin function

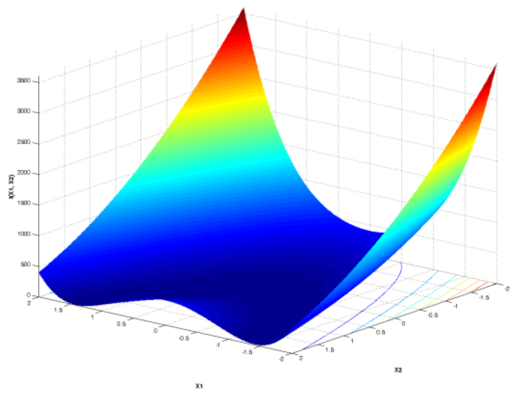


Fig.8. Rosenbrock function

6. RESULTS AND DISCUSSION

Each experiment ran for 30 runs, and average function values for the best solutions were found and recorded. Mean and standard deviations of function values obtained by DE (Krink, etc., 2004), EA (Krink, etc., 2004), PSO (Krink, etc., 2004), BA (Pham, etc., 2006) and PBA under the same conditions are given in Table 2. Values less than E-12 were reported as 0. For f_1 and f_2 functions, DE, EA BA and PBA found the optimum value within the given cycle duration, while PSO could not. For f_3 and f_4 functions, while DE and PBA showed equal performance, it found that optimum, EA, PSO and BA demonstrated relatively worse performance. For the f_5 function, PBA produced the best results. As seen from results presented in Table 3, the PBA produced the best performance amongst all algorithms considered in the present investigation.

Table 3

The results obtained by DE, EA, PSO, BA and PBA

Functions		f_1	f_2	f_3	f_4	f_5
Methods		Schaffer	Sphere	Griewank	Rastrigin	Rosenbrock
DE	Mean	0	0	0	0	35.32
	Std	0	0	0	0	0.27
EA	Mean	0	0	6.24E-3	32.67	79.82
	Std	0	0	1.38E-3	1.94	10.45
PSO	Mean	4.53E-3	2.5113E-8	1.55	13.12	5142.45
	Std	9.00E-4	0	6.70E-2	1.45	2929.47
BA	Mean	0	0	88.27	0	48.65
	Std	0	0	5.18	0	0.43
PBA	Mean	0	0	0	0	10.71
	Std	0	0	0	0	1.70

In order to analyze its behavior, PBA was run using different population (colony) sizes and Table 3 values. As shown in Table 4, best function value means obtained using different colony sizes were 25, 50, 75 and 100 for all functions presented. The progress of mean best values presented in Table 4 is shown in Figs. 9 to 13. From Table 4 and Figs. 9 to 13, it can be concluded that, as population size increases, the algorithm produces better results. However, once the colony size exceeds 75, any increment in fitness value will not significantly improve PBA algorithm performance. For the test problems carried out in this work, a colony size of 75 to 100 can provide an acceptable convergence speed for the search.

Table 4

Mean of function values obtained by PBA under different colony sizes

		Colony sizes			
		25	50	75	100
Functions	Schaffer	1.46E-3	0	0	0
	Sphere	0	0	0	0
	Griewank	2.04E-9	0	0	0
	Rastrigin	6.64E-8	2.92E-11	0	0
	Rosenbrock	86.75	44.04	20.31	10.71

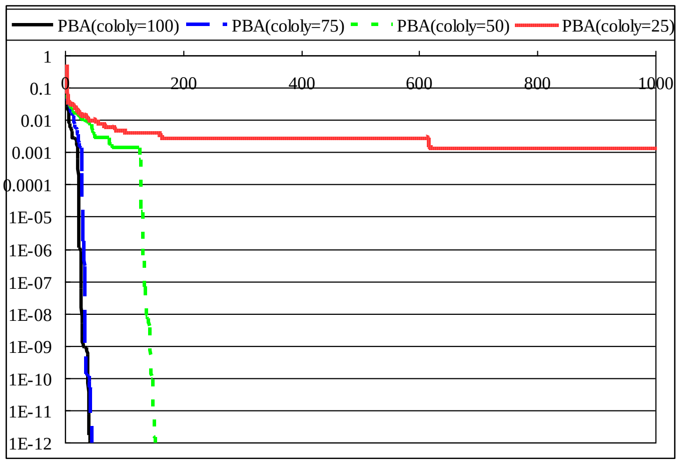


Fig.9. Evolution of mean best values for Schaffer function on different colony sizes

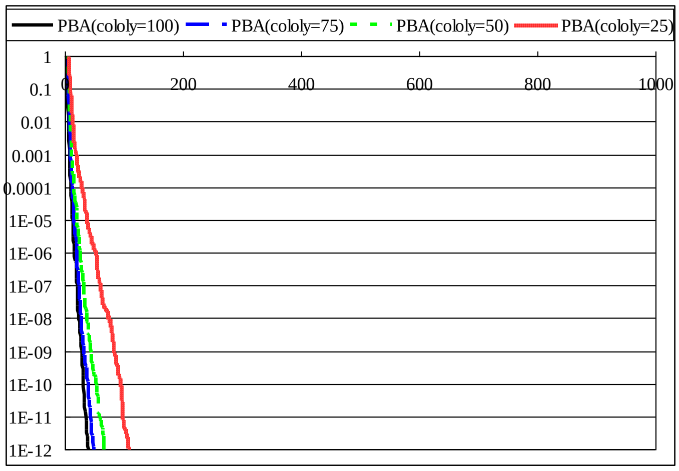


Fig.10. Evolution of mean best values for Sphere function on different colony sizes

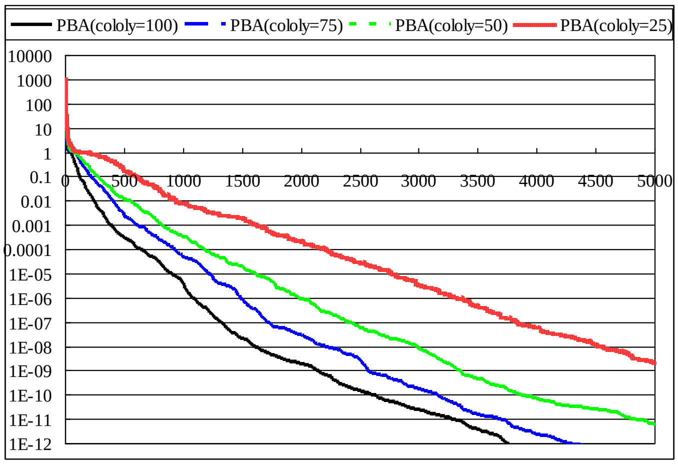


Fig.11. Evolution of mean best values for Griewank function on different colony sizes

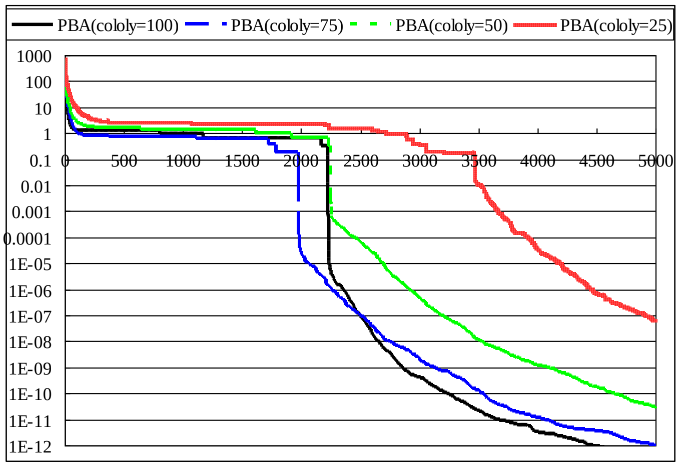


Fig.12. Evolution of mean best values for Rastrigin function on different colony sizes

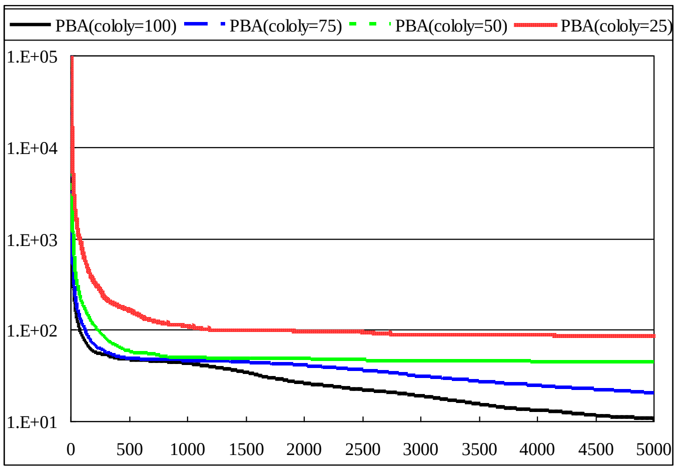


Fig.13. Evolution of mean best values for Rosenbrock function on different colony sizes

Results of colony sizes ranging from 75 to 100 can provide an acceptable convergence speed for search. In order to analyze PBA behavior, this paper has adapted 30 runs with different elite and best bee PSO iteration sizes (*Peitr*, *Pbitr*). In Table 5, the mean of best function values with different elite and best bees PSO iteration sizes vary as (15, 9), (30, 18) and (60, 36) for the presented Rastrigin and Rosenbrock function. Progress of the mean best values presented in Table 5 is illustrated in Figs. 14 to 17. From Table 5 and Figs. 14 to 17, it can be concluded that, during the period in which elite and best bee PSO iterations increase until (30, 18), the algorithm produces better results. However, after a sufficient value for iteration size exceeds (30, 18), the fitness value does not improve, but rather worsens. For test problems carried out in this work, elite and best bee PSO iteration sizes of (15, 9) to (30, 18) can provide an acceptable convergence speed for search.

Table 5

Mean of function values obtained by PBA under different PSO iteration sizes

PSO iteration sizes		(8, 5)	(15, 9)	(30, 18)	(60, 36)
Colony size	Functions				
75	Rastrigin	1.18E-7	0	1.00	4.68
	Rosenbrock	193.61	20.31	21.53	37.14
100	Rastrigin	3.90E-7	0	1.49	8.31
	Rosenbrock	91.10	10.71	10.20	31.83

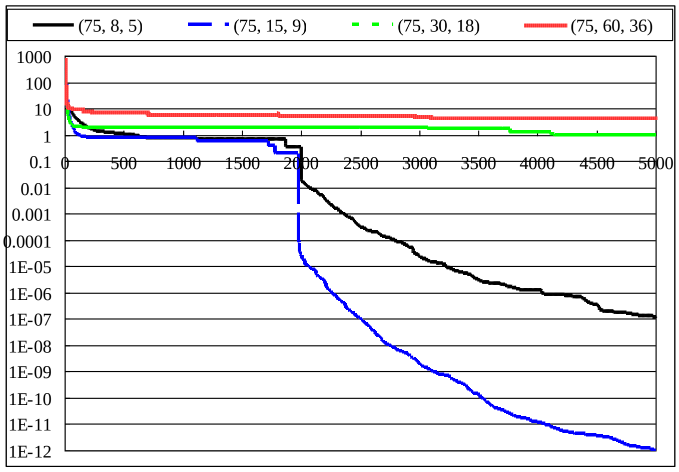


Fig.14. Mean best values for Rastrigin function on 75 colony size and different PSO iteration sizes

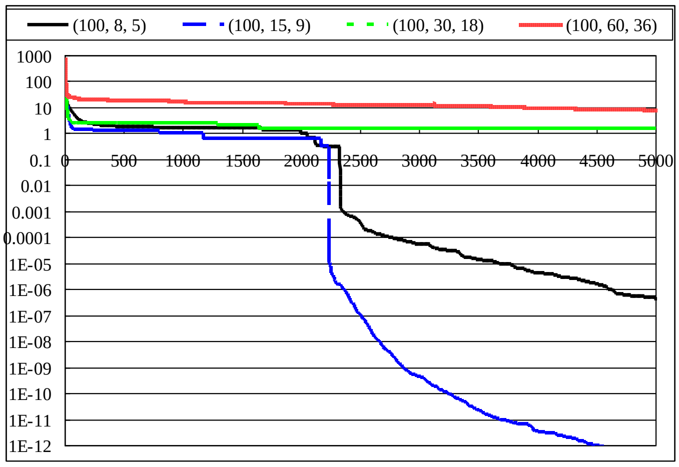


Fig.15. Mean best values for Rastrigin function on 100 colony size and different PSO iteration sizes

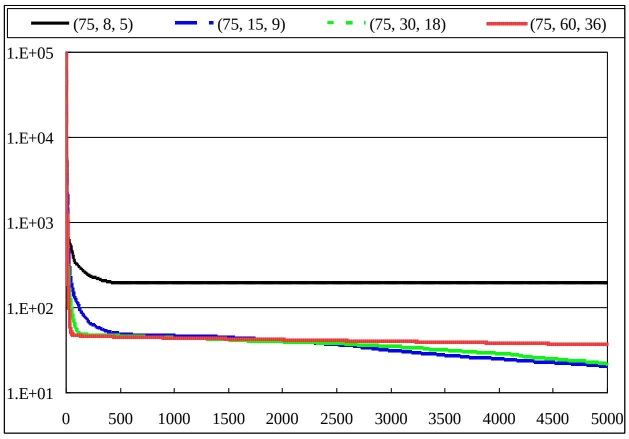


Fig.16. Mean best values for Rosenbrock function on 75 colony size and different PSO iteration sizes

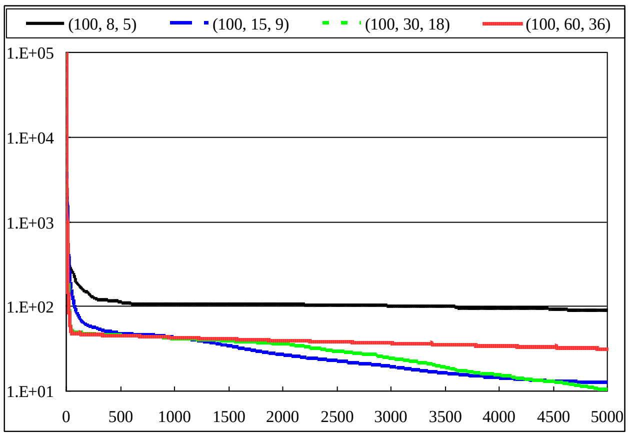


Fig.17. Mean best values for Rosenbrock function on 100 colony size and different PSO iteration sizes

7. CONCLUSION

In the previous section, the performance of the particle bee algorithm (PBA) was compared with differential evolution (DE), evolutionary algorithm (EA), particle swarm optimization (PSO), and bee algorithm (BA) in terms of both multi-dimensional and multimodal numeric problems. In terms of f_1 and f_2 functions, DE, EA BA and PBA identified the optimum value within the given cycle duration, while PSO could not. For f_3 and f_4 functions, while DE and PBA showed equal performance and found the optimum, EA, PSO and BA demonstrated performance that was relatively poorer than DE and PBA. For the f_5 function, PBA produced the best results. Results show that PBA performs better than the mentioned algorithms on each benchmark numerical function. Behavior of PBA under different control parameter values was also analyzed. Results show that the PBA suggests a colony size range of 75 to 100 and a PSO iteration size of (15, 9) to (30, 18) in order to provide an acceptable convergence search speed.

References

1. Basturk B. and Karaboga D., “An Artificial Bee Colony (ABC) Algorithm for Numeric Function Optimization,” *IEEE Swarm Intelligence Symposium 2006*, Indianapolis, Indiana, USA (2006).
2. Bonabeau E., Dorigo M., and Theraulaz G., Swarm Intelligence: From Natural to Artificial Intelligence, *Oxford University Press*, New York (1999).
3. Cheng M.Y. and Lien L.C., “A Hybrid AI-based Particle Bee Algorithm (PBA) for Benchmark Functions and Facility Layout Optimization,” *Journal of Computing in Civil Engineering*, Vol.26, No.5, pp.612-624 (2012).
4. Dorigo, M., “Optimization, Learning and Natural Algorithms,” *Ph.D. Thesis, Politecnico di Milano*, Italy (1992).
5. Eberhart, R., Shi Y., and Kennedy J., Swarm Intelligence. Morgan Kaufmann, *San Francisco* (2001).
6. Holland J.H., Adaptation in Natural and Artificial Systems, *University of Michigan Press*, Ann Arbor, MI (1975).
7. Karaboga D. and Akay B., “A comparative study of Artificial Bee Colony algorithm,” *Applied Mathematics and Computation*, Vol.214, pp.108-132 (2009).
8. Kennedy J. and Eberhart R.C., “Particle swarm optimization,” *In Proceedings of the 1995 IEEE International Conference on Neural Networks*, Vol.4, pp.1942-1948 (1995).
9. Korenaga T., Hatanaka T. and Uosaki K., “Improvement of Particle Swarm Optimization for High-Dimensional Space,” *2006 SICE-ICASE International Joint Conference* (2006).
10. Krink T., Filipic B., Fogel G.B. and Thomsen R., “Noisy optimization problems—a particular challenge for differential evolution?,” *Proceedings of 2004 Congress on Evolutionary Computation, IEEE Press*, Piscataway, NJ, pp.332-339 (2004).
11. Li, X. L., “A new intelligent optimization-artificial fish swarm algorithm,” *Ph.D. Thesis, Zhejiang University of Zhejiang*, China (2003).
12. Lien L.C. and Cheng M.Y., “A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization,” *Expert Systems with Applications*, Vol.39, No.10, pp.9642-9650 (2012).
13. Lien L.C. and Cheng M.Y., “Particle Bee Algorithm for Tower Cranes Layout with Materials Quantity Supply and Demand Optimization,” *Automation in Construction*, Vol.45, No.9, pp.25-32 (2014).
14. Ozbakir L., Baykasog A. and Tapkan P., “Bees algorithm for generalized assignment problem,” Applied Mathematics and Computation, Vol.215, pp. 3782-3795 (2010).

15. Parsopoulos, K. E., & Vrahatis, M. N., “Parameter selection and adaptation in unified particle swarm optimization,” *Mathematical and Computer Modeling*, Vol.46, No.1, pp.198-213 (2007).

16. Pham D.T., Koc E., Ghanbarzadeh A., Otri S., Rahim S. and Zaidi M., “The bees algorithm - a novel tool for complex optimization problems,” *In Proceedings of the Second International Virtual Conference on Intelligent Production Machines and Systems*, pp.454-461 (2006).

17. Price K.V., Storn R.M. and Lampinen J.A. (Eds.), Differential Evolution: A Practical Approach to Global Optimization, *Springer Natural Computing Series* (2005).

18. Tsai H. C., “Predicting strengths of concrete-type specimens using hybrid multilayer perceptions with center-unified particle swarm optimization,” *Expert Systems with Applications*, Vol.37, pp.1104-1112 (2010).

19. Yang X.S., “Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms,” *Lecture Notes in Computer Science*, Vol.3562, pp.317-323 (2005).