Available online at: http://publikasi.mercubuana.ac.id/index.php/ijiem

IJIEM (Indonesian Journal of Industrial Engineering & Management)

ISSN (Print) : 2614-7327 ISSN (Online) : 2745-9063

Foraging Bee Optimization Algorithm

Ebun Phillip Fasina*, Babatunde Alade Sawyerr, Shuaibu Babangida Alkassim
Department of Computer Science, University of Lagos, Lagos, Nigeria

ARTICLE INFORMATION

Article history:

Received: 3 May 2023
Revised: 2 June 2023
Accepted: 4 June 2023

Category: Research paper

Keywords:
Swarm intelligence
Nature-inspired metaheuristics
Bee-inspired optimization algorithm
Numerical optimization
Particle swarm optimization

DOI: 10.22441/ijiem.v4i2.20275

A B S T R A C T

Honeybees feed on pollen and nectar from flowers. Nectar to meet their energy requirements and pollen for protein and other vital nutrients. The act of searching for these flowers by honeybees is called foraging. The foraging behaviour of bees depends on the profitability of nectar and pollen sources as well as the needs of the colony. This behaviour is modelled by the Foraging Bee Optimization Algorithm (FBA) as metaphor for optimization. After initialization, the algorithm loops through three phases based on bees' foraging behaviour –work, withdraw, and waggle (3W). Flowers are initialized randomly in the problem space. During the waggle phase, bees are recruited to flowers with profitable nectar sources. In the work phase, new flowers are discovered and memorized by bees. In the withdraw phase bees remove unprofitable flowers from collective memory and recalibrate for recruitment. The proposed FBA is tested on three unimodal and twelve multimodal benchmark functions. The result is compared with two other state-of-the-art swarm intelligence algorithms, Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO). Analysis of comparison results shows FBA to be highly competitive, outperforming PSO on all benchmarks and matching ABC in overall performance.

*Corresponding Author
Ebun Fasina
E-mail: efasina@gmail.com

1. INTRODUCTION

The study of the behavior of social organisms as a swarm in and outside their colonies led to Swarm Intelligence (SI) (Eberhart, Shi, & Kennedy, 2001; Janaki & Geethalakshmi, 2022; Selvaraj & Choi, 2020). SI is a discipline in computer science that mimics the intelligence displayed by social organisms (Kaswan, Dhatterwal, & Kumar, 2023; Schumann, 2020). This intelligence can be self-learning, healing, or optimizing. Researchers model and create algorithms based on this intelligence. These algorithms are classified as Nature-Inspired or Swarm Intelligence Optimization algorithms or metaheuristics and have been applied to solve a diverse range of problems (Fakhermand & Derakhshani, 2023; Tzanetos & Dounias, 2020; Engelbrecht, 2007; Alizadehsani, et al., 2023; Altshuler, 2023; Shahzad, et al., 2023; Kumar, Chatterjee, Payal, & Rathore, 2022; Cruz, Maia, & de Castro, 2021).

Nature-Inspired algorithms find approximate solutions to optimization problems, the solution can be local or global optimum depending on the set of constraints the optimization problem

is subjected to. An optimization problem requires an objective function that may be constrained or unconstrained to be maximized or minimized. Optimization algorithms invoke the objective function to determine the fitness of a large and varied selection of solutions to determine the best or near optimum. Optimization techniques are mostly applied to minimize cost or error, maximize profit, and find optimal designs for engineering problems or provide optimal decisions for operational and management problems.

Various natured-inspired algorithm has been proposed among which are the Particle Swarm Optimization (PSO) by Kennedy and Eberhart (Kennedy J. and Eberhart, 1995) which is inspired by simulation studies of the social behavior found in schools of fish and flocks of birds. Bee Colony Optimization (BCO) by (Teodorovic & Dell'orco, 2005), Bee Algorithm (BA) by (Pham, et al., 2005), Artificial Bee Colony by (Karaboga, 2005) are all inspired the foraging behavior of bee colonies. Genetic recombination and natural selection inspired the Generic Algorithm (GA) proposing by (Holland, 1975). Studies of ant colonies resulted in the Ant Colony Optimization (ACO) algorithm by (Dorigo, Colorni, & Maniezzo, 1991). Differential Evolution (DE) was proposed by (Storn & Price, 1997), and Glowworm Swarm Optimization was proposed by (Krishnanand & Ghose, 2005). GSO mimics the behavior of luminescent glowworms in nature.

In this work a new algorithm called FBA that is inspired by the foraging behavior of bees is proposed and implemented to improve the speed of convergence of bee-inspired algorithms, avoid premature convergence as well as balance exploitation with exploration.

2. LITERATURE REVIEW

I. Foraging Bee in Nature

Honeybees are social insects or organisms that live together in well-organized colonies and can perform complex tasks in reasonable time with ease. These tasks include controlling the environment, division of labor, defense of nest and queen, nest construction, communications, and foraging for food. The process of foraging for food involves scouting, collection of pollen and nectar from flowers, and conveyance of pollen and nectar to the colony. Bees in charge of foraging are called foragers. Each forager modulates its behaviour in relation to the profitability of the nectar source – the more profitable the source, the higher the intensity of foraging activity around the source, the more repetitive and dancing (or waggle) at the nest pointing to the source, and the lower the probability of abandoning the source. Without comparing sources, bee individually calculate the absolute profitability of a source. The collective nectar and pollen source selection by a colony of bees is decentralized; it is a process of natural selection where foragers from more profitable nectar sources continue to visit these sources over a long period of time and eventually recruit bees from less profitable sources. In a typical foraging season, bees collect roughly 20 – 30 kg of pollen and 125kg nectar which translate to between 1,125,000 and 4,000,000 visits to flowers.

II. Bee Colonies as Metaphors for Swarm Intelligence Algorithms

Agents in the Bee Algorithm (BA) first proposed by (Pham, et al., 2005) combined randomized search of the problem space with neighborhood search in promising regions of this space. BA is complex and can easily be trapped in a local optimum. The Artificial Bee Colony (ABC) algorithm proposed by (Karaboga, 2005) is less complex when compared with previous bee optimization algorithms (Bolaji, Khader, Al-Betar, & Awadallah, 2013) but converges poorly. (Sato & Hagiwara, 1997) reformulated the Genetic Algorithm (GA) to develop a new algorithm called the Bee System (BS). BS performs global search using GA operators and then improves on local search by introducing new operators such as concentrated crossover and the pseudo-simplex method.

The mating behavior of bees is the inspiration for the Mating Bee Optimization MBO algorithm (Abbass, 2001). MBO algorithm begins with one queen with no relatives, to a colony of relatives with a single queen or multiple queens. MBO has been modified several to form a new algorithms such as the Honey Bee Optimization (HBO) algorithm by (Curkovic & Jerbic, 2007), Honey Bees Mating

Optimization (HBMO) algorithm by (Haddad, Afshar, & Mariño, 2006) and the Fast Marriage in Honey Bees Optimization (FMHBO) algorithm by (Yang, Chen, & Tu, 2007).

(Gao, Liu, & Huang, 2012) modified the ABC algorithm in order to improve its exploitation. The new algorithm called ABC/Best searches only around the fittest bee based on the last best solution. They employed a chaotic system and opposition-based learning for improving the speed of global convergence.

(Mathlouthi & Bouamama, 2016) integrated a centralized and distributed technique called a local optimum detector to an algorithm inspired by marriage in honeybees. The local detector enhanced finding the local optimum. (Li & Yang, 2016) proposed a variant of ABC. They introduced a memory mechanism that aids artificial bees by memorizing their best foraging experience so far. (Pan, 2016) hybridized ABC and GA to develop a self-adaptive algorithm with a dual population of independently evolving bees that exchange information through information entropy that ensures diversity and accelerates convergence.

(Pan, 2016) proposed a hybrid, self-adaptive genetic-bee colony algorithm based on information entropy. The algorithm evolved two populations of bees independently but allowed the exchange of information between bees in the two populations using entropy to maintain population diversity and accelerate the evolution process. Under analysis it was found that this strategy accelerated the emergence of fitter individuals by competition between the populations performs better in complex function optimization problems.
(Aslan, Karaboga, & Badem, 2020) modeled the complex behavior of foraging bees in detail – how they pass through the dance area and how long they performed their dance to attract onlooker bees – then adapted it to ABC to develop a new variant of ABC, termed the intelligent forager forwarding ABC (iff-ABC). They analyzed the contribution of the intelligent forager forwarding strategy on the performance of ABC algorithms by evaluating its performance on the CEC benchmark suite and comparing it with the performance of different variants of ABC. The results obtained showed that an intelligent forager forwarding strategy significantly improves the quality of final solutions and the convergence speed of ABC algorithms.

(Chen, Tianfield, & Du, 2021) proposed a novel bee-foraging learning PSO (BFL-PSO) algorithm that is inspired by the search mechanism of the artificial bee colony algorithm. The proposed BFL-PSO has three different search phases, namely: employed learning, onlooker learning and scout learning. The employed learning phase is the one-phase-based PSO search, while the onlooker learning phase exploits the region around promising solutions, and the scout learning phase introduces new diversity by re-initializing stagnant particles. The proposed BFL-PSO is evaluated on the CEC2014 benchmark suite, and compared with state-of-the-art PSO and artificial bee colony algorithms The experimental results show BFL-PSO to be competitive in performance and the accuracy of its solutions.

It is helpful to study and compare various versions of bee inspired metaheuristics to enable the selection of these algorithms in the optimization tasks and the refinement and development of new variants. (Solgi & Loáiciga, 2021) identifies seven basic or root algorithms applied to solve continuous optimization problems, namely: Bee System (BS), Mating Bee Optimization (MBO), Bee Colony Optimization (BCO), Bee Evolution for Genetic Algorithms (BEGA), Bee Algorithm (BA), Artificial Bee Colony (ABC), and Bee Swarm Optimization. They ranked these algorithms by performance and convergence efficiency and found ABC, BEGA, and MBO to be the most efficient. They discussed the strengths and shortcomings of each algorithm and explained the variations observed in the convergence rate of these algorithms.

3. THE FORAGING BEE OPTIMIZATION ALGORITHM (FBA)

The Foraging Bee (Optimization) Algorithm (FBA) is inspired by the foraging behaviour of bees for pollen and nectar, and the collective natural selection of more profitable nectar sources over poor ones. The FBA algorithm is

developed. In FBA, the colony consists of bees, termed foragers, who scout for flowers that are rich sources of pollen and nectar in a patch of the problem space in the *work phase*, then return to the colony during the *withdraw phase* to communicate their findings using dance in the *waggle phase*. The FBA pseudocode is listed below as follows:

Bee

A bee b_i is modeled by the tuple $B = B(x_B, f_B, D, P)$ where x_B is the vector representing the current position of the bee, $f_B \leftarrow f(x_B)$ is the fitness of the current position of the bee, D is the direction of the bee and P is the patch in which the bee is initialized. Each bee makes a foraging move in time $t+1$ in dimension j as follows:

$$x_j(t+1) = x_j(t) + pr_1\left(d_j^+\{U_j - x_j(t)\} + d_j^-\{x_j(t) - L_j\}\right) \quad (1)$$

where r_1 is a random number between 0 and 1, U_j *and* L_j are the upper and lower bounds in dimension j of patch P, p is the propensity of the bee. The direction vector D is a unit vector indicating the current direction of the foraging bee. Bees make decisions before moving in direction D by determining the direction $d_j^{\pm}$ to move in each dimension j using the random variable $r_2 \sim U(0,1)$. Assume that the bee is moving in direction d_j^+ in time t. The decision to continue in direction d_j^+ is determined by

$$r_2 < \frac{|a_j - x_j|}{U_j - L_j} \quad (2)$$

where $A = (a_1, a_2, \ldots, a_n)$ is the bee attractor in each patch. If (2) is true and c_j is in direction of U_j, then $d_j^+ = 1$ and $d_j^- = 0$ otherwise $d_j^- = 1$ and $d_j^+ = 0$.

Algorithm 1: Foraging Bee Optimization Algorithm

1 **set** the following parameters
- B_{pop} is the population of bees
- M is the *minimum* population of flowers
- N is the *minimum* population of newly discovered flowers
- K number of scouts added as recruits during each waggle phase
- P_{prob} is the search space
- β is fraction of resource rich flowers for estimating the attractor of a patch
- p is the propensity of bees when exploring patches

2 **initialize** M flowers in patches
set f_T as the fitness of the fittest flower
initialize bees randomly in patch P
termcond ← **false**
$n \leftarrow 1$, $k \leftarrow 0$

3 **while true**
- // WORK PHASE
- **for** $i = 1$ **to** T
  - **move** bee b_i
  - **if** $f(b_i) < f_T$ **mark** b_i with flower F_{M+n} and **increment** n
- **if** $n < N$ **then continue**
- // WITHDRAW PHASE
- *termcond* ← GET-TERMCOND( )
- **if** *termcond* **then**
  - **return** fittest flower as optimum
- $n \leftarrow 1$
- **set** f_T as the fitness of the fittest flower
- // WAGGLE PHASE
- **select** best M flowers in P_{prob}
- Estimate promising patch using selected flowers P_{best}
- Determine the location of attractors in each patch.
- **increment** k
- **initialize** k bees (recruits) randomly in P_{best}
- **initialize** other bees $B_{pop} - k$ (scouts) in P_{prob}

The propensity p determines how bees explore or exploit a patch. Lower values of p favors exploitation over exploration. The continuous reduction in the spatial dimensions of P_{best} allows the exploitation of promising patches by recruits while scouts continue to explore the entire problem space. The bee search equation guides the bees to forage only within the patch in which they are initialized making exploration and exploitation explicit processes guided by patches P_{prob} and P_{best}.

Flower

A flower F is modeled by the tuple $F = F(x_F, f_F)$, where x_F is the vector representing the current position of the flower F and f_F is its fitness. The lower value of f_F the richer the flower as a source nectar and pollen to bees.

Patch

Patches are modeled by the tuple $P = P(L, U, A)$ where L is the vector that represents the lower limit of the patch in all dimensions, U is the vector that represents the upper limit of the patch in all dimensions and A is the bee attractor. P_{prob} is initialized with $M + N$ or more flowers while P_{best} is estimated with M best flowers. The point attractor of bees in a patch is the centroid the best flower f_T and a fraction $\beta \cong 0.5$ of the other flowers in the patch. Candidate flowers for bee attractor computation selected using the roulette operator []. Unlike GA flowers are selected without replacement, i.e., a candidate flower cannot be selected more than once. It is important that the point attractor of bees in patches P_{prob} and P_{best} are not coincident at the early stages of search. Observe that all points in P_{best} are interior points of P_{prob}.

Foraging Bee Algorithm

The flowchart in Fig. 1 highlight phases FBA. It begins with the initialization of search parameters and objects such as bees, patches, and flowers. This is followed by the *work* phase where scout bees and recruits search for new resource rich flowers. During the *withdraw* phase critical parameters are reset and the GET-TERMCOND method determines if an approximate solution has been found or the maximum number of objective function evaluation has been exceeded. If the algorithm does not stop it enters the *waggle* phase where information is shared; P_{best} is initialized or recalibrated and scouts are recruited to exploit the patch. The algorithm repeats the work, withdraw and waggle phases until it terminates in the withdrawal phase.

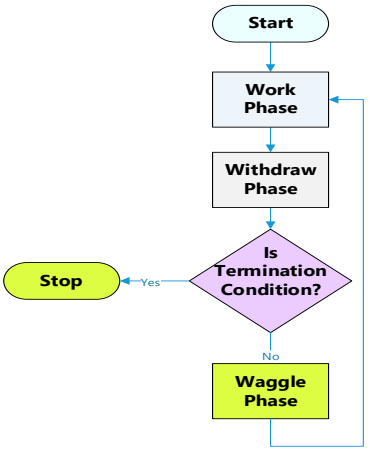

Fig. 1. Flowchart of FBA

4. RESULT AND DISCUSSION

The FBA algorithm was run on standard benchmark test function; these functions were presented in Table 1 as equations (3) to (17) and their properties a tabulated in Table 2. They were carefully chosen to test FBA's capacity to solve problems with diverse properties and varying levels of difficulty. f_1 to f_3 are simple unimodal functions while f_4 to f_{15} are multimodal functions with local minima ranging from a few hundred to millions. The performance of FBA on each test function is benchmarked with 30 trials of 50000 function evaluations.

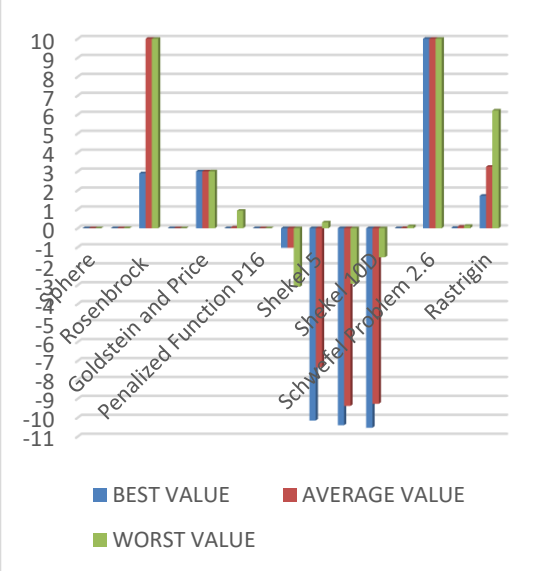

Fig. 2. Graph of FBA results

The overall performance based on the best, average, and worst-case error rates, standard deviation, and success rate of each test function

is tabulated in Table 3 and shown graphically in Fig. 2. The success rate of each function is also shown in Fig. 5. The success of any run is determined by an error of at least four leading zeros (E-04).

Table 1. Benchmark test functions

1)	Sphere function $f_1(x) = \sum_{i=1}^{n} x_i^2$	(3)
2)	Schwefel P2.22 function $f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	(4)
3)	Rosenbrock's function $f_3(x) = \sum_{i=1}^{n-1} \{100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\}$	(5)
4)	Ackley F1 $f_4(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	(6)
5)	Goldstein-Price $f_5(x) = \{1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\} \times \{30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\}$	(7)
6)	Penalized Function P8 $f_6(x) = \frac{\pi}{x}\left\{10 \sin^2(\pi y_i) + \sum_{i=1}^{D-1} \{1 + 10 \sin^2(\pi y_{i+1})\} + (y_d - 1)^2\right\} + \sum_{i=1}^{D} \mu(x_i, 10,100,4)$	(8)
7)	Penalized Function P16 $f_7(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{n-1} (x_i - 1)^2\{1 + 10 \sin^2(3\pi x_{i+1})\} + (x_d - 1)^2\{1 + 10 \sin^2(2\pi x_D)\}\right\} + \sum_{i=1}^{D} \mu(x_i, 5,100,4)$	(9)
8)	Schaffer's F6 function $f_8(x) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^{n} x_i^2}\right) - 0.5}{\{1 + 0.001(\sum_i^n x_i^2)\}^2}$	(10)
9)	Shekel 5 function $f_9(x) = -\sum_{i=1}^{5} \frac{1}{\sum_{j=1}^{4}(x_i - a_{ij})^2 + c_i}$	(11)
10)	Shekel 7 function $f_{10}(x) = -\sum_{i=1}^{7} \frac{1}{\sum_{j=1}^{4}(x_i - a_{ij})^2 + c_i}$	(12)
11)	Shekel 10 function	

$$f_{11}(x) = -\sum_{i=1}^{10} \frac{1}{\sum_{j=1}^{4}(x_i - a_{ij})^2 + c_i} \quad (13)$$

$$A = [a_{ij}] = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix} \qquad C = [c_i] = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}$$

12) Six-Hump Camelback

$$f_{12}(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \quad (14)$$

13) Schwefel P2.6 function

$$418.9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right) \quad (15)$$

14) Griewank's function

$$f_{14}(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (16)$$

15) Rastrigin's function

$$f_{15}(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \quad (17)$$

Table 2. Properties of benchmark test functions

	Name	**Feasible Bounds**	n	**Optimum,** x^*	$f(x^*)$
f_1	Sphere	$[-100, 100]^n$	5	0^n	0
f_2	Schwefel P2.22	$[-500, 500]^n$	5	420.9687^n	0
f_3	Rosenbrock's	$[-100, 100]^n$	5	1^n	0
f_4	Ackley's F1	$[-32.768, 32.768]$	5	0^n	0
f_5	Goldstein-Price	$[-2, 2]$	2	$(0, -1)$	0
f_6	Penalized F8	$[-50, 50]$	5	-1^n	0
f_7	Penalized P16	$[-50, 50]$	5	1^n	0
f_8	Schaffer F6	$[-100, 100]^n$	2	0^n	0
f_9	Shekel 5	$[0, 10]^n$	4	4.0^n	-10.1499
f_{10}	Shekel 7	$[0, 10]^n$	4	4.0^n	-10.3999
f_{11}	Shekel 10	$[0, 10]^n$	4	4.0^n	-10.5319
f_{12}	Six-Hump Camel	$[-5, 5]^n$	2	(-0.0898, 0.7126), (0.0898, -0.7126)	-1.0316
f_{13}	Schwefel P2.6	$[-500, 500]^n$	5	420.9687^n	0
f_{14}	Griewank	$[-600, 600]^n$	5	0^n	0
f_{15}	Rastrigin	$[-5.12, 5.12]$	5	0^n	0

Table 3. The summary results obtained by the FBA algorithms for 30 runs

Func.	Best Value	Average Value	Worst Value	Std. Dev.	Success Rate
f_1	1.0686E-119	1.7687E-16	5.2875E-15	9.6525E-16	100
f_2	4.6843E-33	3.7497E-04	7.0707E-03	1.4766E-03	93.33
f_3	2.8994E+00	1.8798E+04	2.2862E+04	4.4271E+03	0
f_4	0.0000E+00	7.1304E-08	2.1232E-06	3.8756E-07	100
f_5	-9.5923E-14	5.4712E-05	1.6414E-03	2.9967E-04	96.67
f_6	2.1903E-11	3.8512E-02	9.2391E-01	1.6801E-01	43.33
f_7	1.4096E-14	6.2899E-05	1.5010E-03	2.7692E-04	96.67
f_8	3.3695E-13	3.7001E-04	2.4989E-03	5.6688E-04	90
f_9	-3.8281E-06	2.7995E+00	7.6638E+00	2.8028E+00	33.33
f_{10}	-1.2173E-04	1.0171E+00	6.4562E+00	1.8296E+00	60
f_{11}	-1.2609E-04	1.2683E+00	7.7326E+00	2.2092E+00	66.67
f_{12}	-3.0562E-08	9.4618E-03	1.0871E-01	2.0714E-02	33.33
f_{13}	3.5809E+01	1.4418E+02	2.0573E+02	4.3292E+01	0
f_{14}	2.1281E-02	7.9438E-02	1.2790E-01	2.6691E-02	0
f_{15}	1.7127E+00	3.2470E+00	6.2162E+00	1.1694E+00	0

Fig. 3(a) to (k) show successful runs of FBA on 11 benchmark test functions for which it converges, and successfully returns at least once an approximate solution to the optimum with error rates less the 1E-08. Fig. 4 (a) to (d) on the other hand are unsuccessful runs of FBA on 4 benchmark test functions.

FBA is compared ABC and PSO using T-test. Table 4 shows the mean and standard error of FBA, ABC and PSO on the test function while Table 5 tabulates the results of the T-test and indicates test functions in which the performance of FBA is statistically significant when compared with both ABC and PSO. FBA did not return any success for four (f_3, f_{13}, f_{14}, and f_{15}) benchmark functions out of the fifteen tested on. Three (f_6, f_9, and f_{12}) were below fifty percent while the remaining eight ranges from sixty to hundred percent.

Results in Table 5 show that the comparison between FBA and PSO on all 10 benchmark test functions is statistically significant. Results also show that 7 out of the 10 benchmark tests between FBA and ABC are statistically significant. In nine of the ten statistically significant benchmark test FBA performed better than PSO while in two of the seven statistically significant test FBA performed better than ABC.

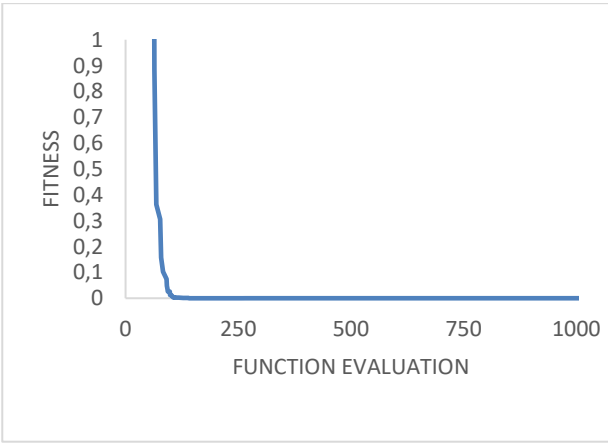

Fig. 3(a). Sphere

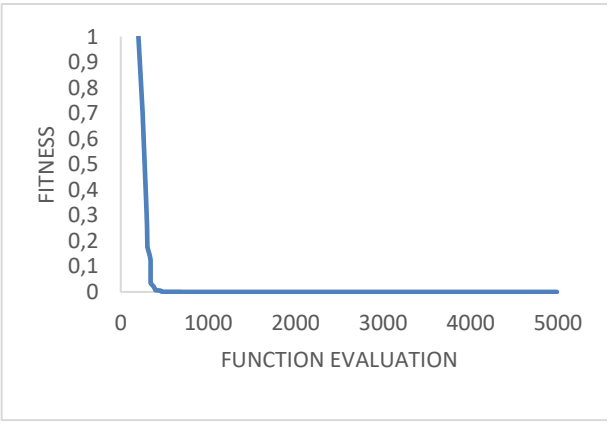

Fig. 3(b). Schwefel P2.22

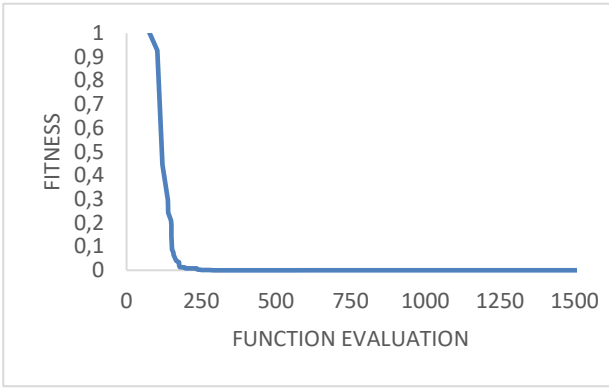

Fig. 3(c). Ackley

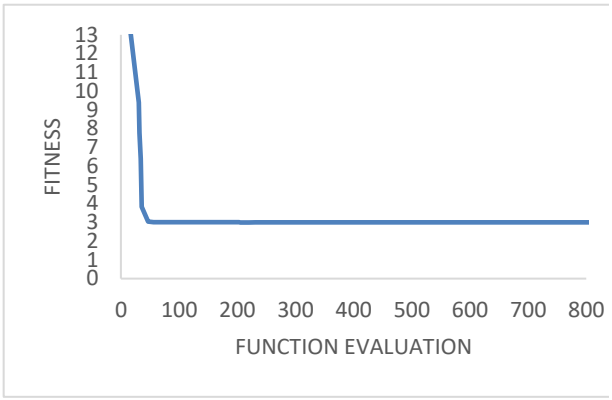

Fig. 3(d). Goldstein-price

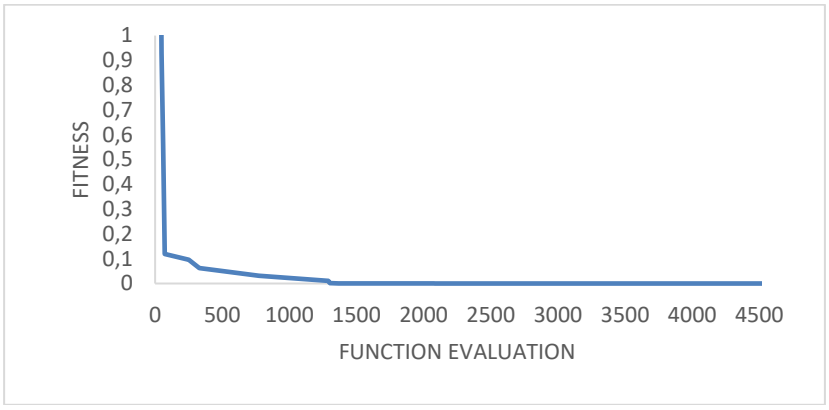

Fig. 3(e). Penalized function P8

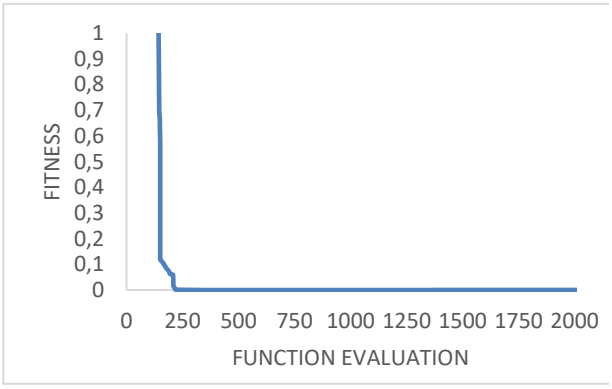

Fig. 3(f). Penalized function P16

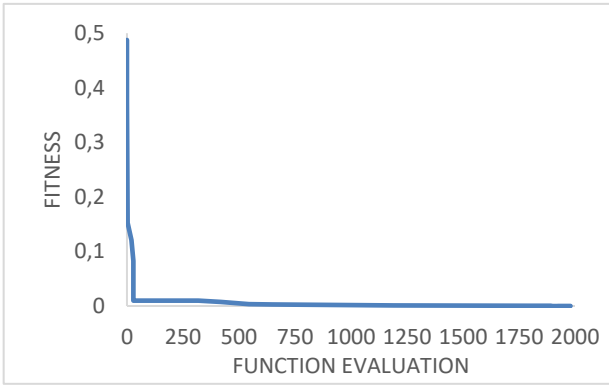

Fig. 3(g). Schaffer F6

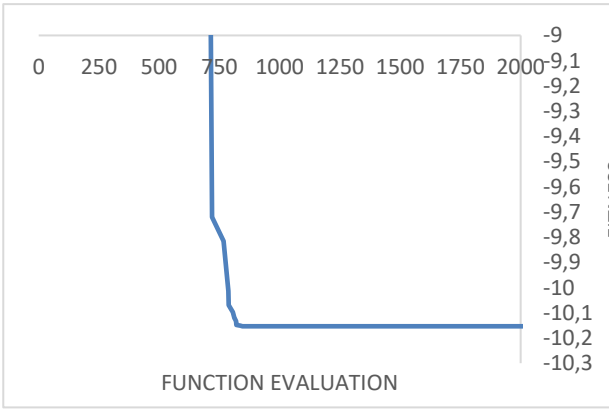

Fig. 3(h). Shekel 5

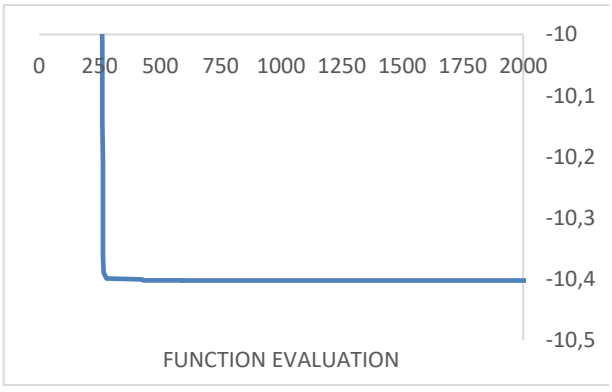

Fig. 3(i). Shekel 7

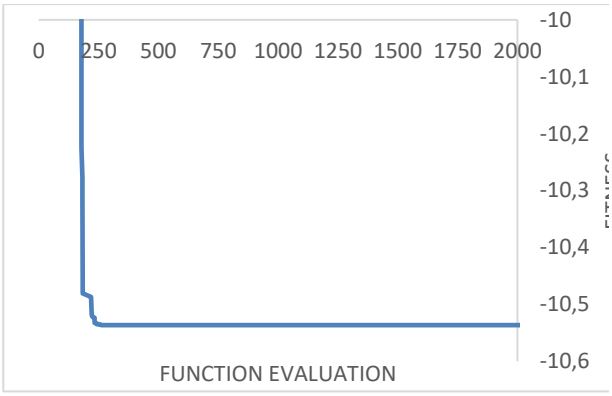

Fig. 3(j). Shekel 10

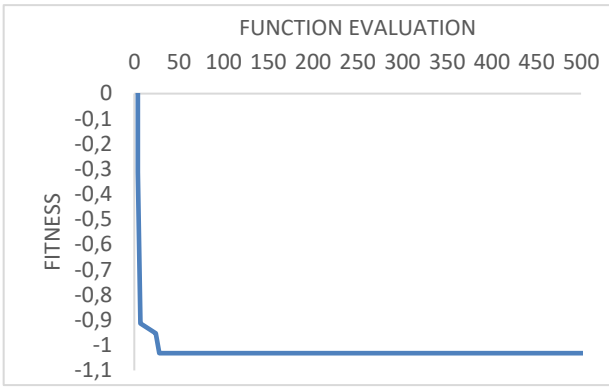

Fig. 3(k). Six-Hump Camel

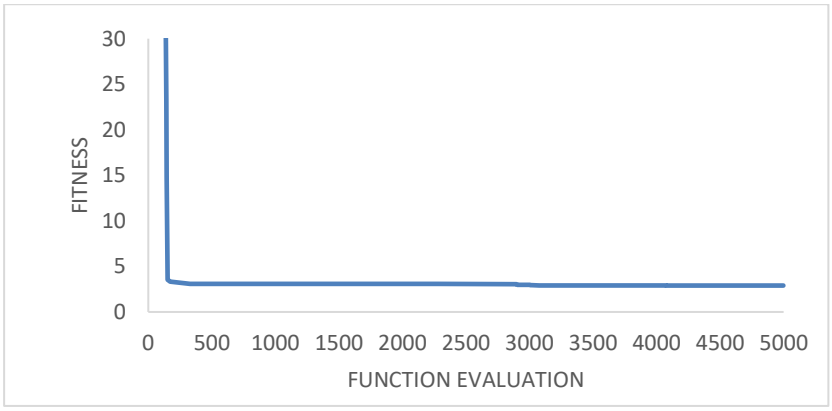

Fig. 4(a). Rosenbrock

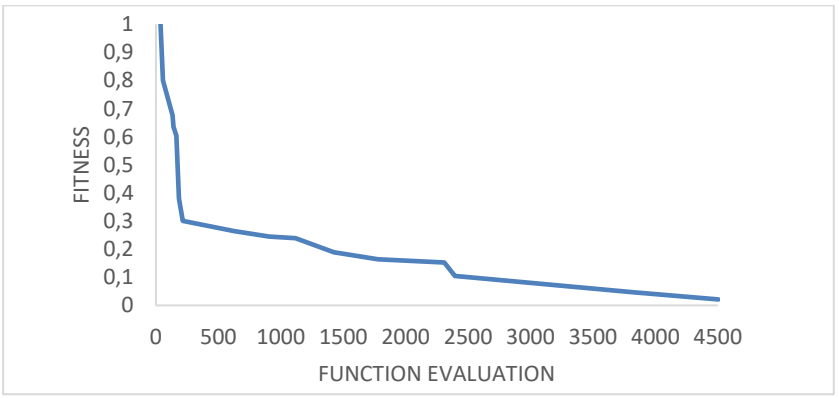

Fig. 4(b). Griewank

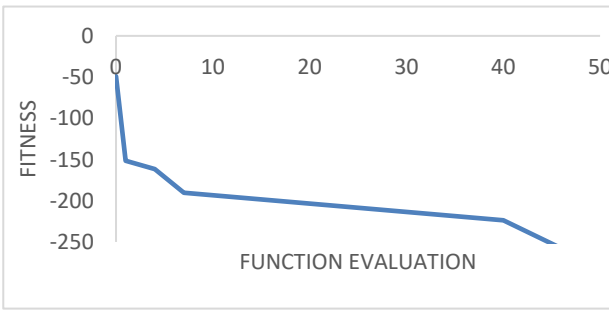

Fig. 4(c). Schwefel P2.6

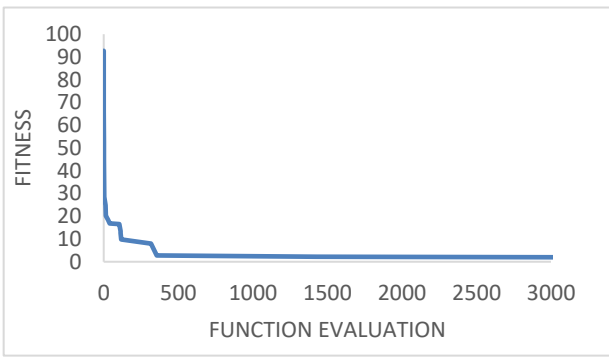

Fig. 4(d). Rastrigin

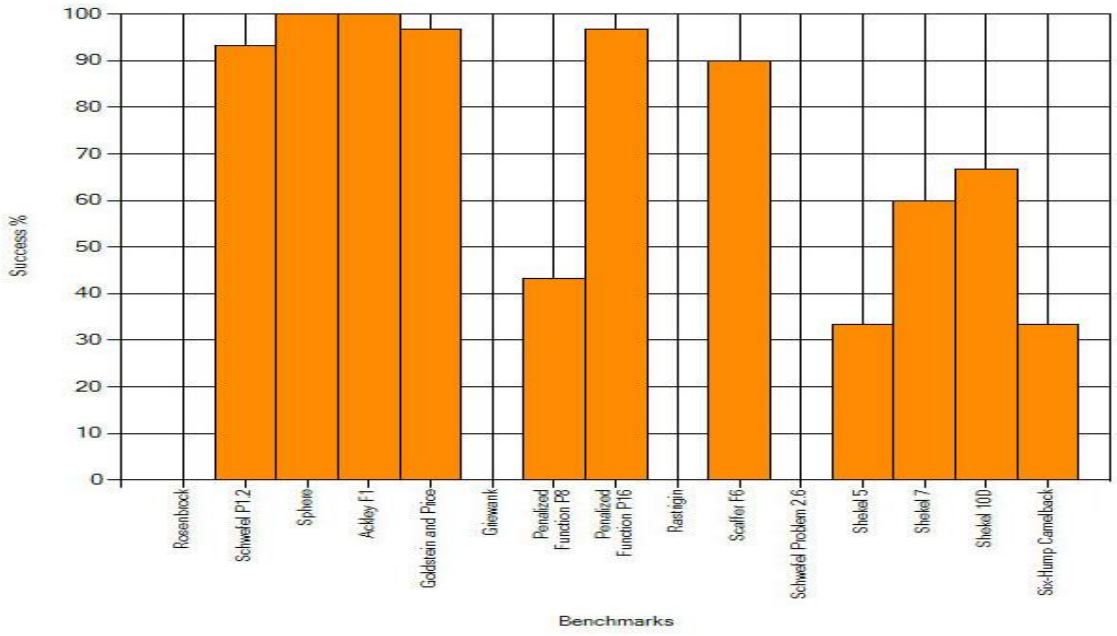

Fig. 5. Histogram of success rates

Table 4: The Comparison between FBA, PSO, and ABC

Function	FBA		ABC		PSO	
	Mean	Std. Error	Mean	Std. Error	Mean	Std. Error
Sphere	1.77E-16	±1.76E-16	6.99E-10	±1.08E-10	2.75E+00	±4.48E-02
Schwefel P2.22	3.75E-04	±2.70E-04	2.36E-06	±1.52E-07	5.45E+00	±1.43E-01
Rosenbrock	1.87E-03	±8.08E+02	3.93E-02	±5.68E-03	5.46E+01	±2.85E+00
Ackley F1	7.13E-08	±7.08E-08	1.02E-05	±4.45E-03	2.02E+01	±8.20E-03
Penalized F. P8	3.85E-02	±3.07E-02	1.60E-11	±3.56E-12	8.86E+00	±3.83E-01
Penalized P16	6.29E-05	±5.06E-05	3.72E-09	±3.27E-10	1.22E-01	±2.10E-03
Schaffer F6	3.70E-04	±1.03E-04	2.07E-03	±7.14E-04	-	-
Shekel 7	-9.38E+00	±3.34E-01	-1.04E-01	±2.88E-16	5.31E+00	±6.05E-06
Shekel 10	-9.26E+00	±4.03E-01	-1.05E-01	±6.19E-15	5.40E+00	±6.19E-06
Griewank	7.94E-02	±4.87E-03	8.73E-09	±2.68E-09	1.01E+00	±3.10E-03
Six-Hump Camel	9.46E-03	±2.07E-02	-	-	1.77E+00	±2.89E-01

Table 5: The T-test between FBA/PSO and FBA/ABC

	T-test FBA/PSO			T-test FBA/ABC		
Function	Value	Critical Value	Significant	Value	Critical Value	Significant
Sphere	61.5223	<0.00001	YES	6.4781	<0.00001	YES
Schwefel P2.22	38.2318	<0.00001	YES	1.3829	0.0885	YES
Rosenbrock	2.2662	0.0154	YES	2.3258	0.0135	YES
Ackley F1	2472.81	<0.00001	YES	0.0023	0.4991	NO
Penalized F. P8	25.0648	<0.00001	YES	1.2555	0.1095	NO
Penalized P16	59.4977	<0.00001	YES	1.2440	0.1116	NO
Schaffer F6	-	-	-	2.7852	0.0046	YES
Shekel 7	-44.8228	<0.00001	YES	3.0444	0.0024	YES
Shekel 10	-36.3861	<0.00001	YES	3.1452	0.0019	YES
Griewank	-525.4461	<0.00001	YES	16.3014	<0.00001	YES
Six-Hump Camel	6.5879	<0.00001	YES	-	-	-

V. CONCLUSION

The Foraging Bee Optimization (FBA) algorithm is a swarm intelligence optimization algorithm, which has a new unique approach inspired by the characteristics and intelligent behavior displayed by the swarm of foraging bees for solving optimization problems. The algorithm mimics bee colonies by organizing search into three phases: work – when bees forage in patches and discover and exploit new resource-rich flowers; withdraw – when bees return to the colony and reset for the next work phase; and waggle – when bees share information about locations containing resource rich flowers.

The algorithm increased the speed of convergence and balance exploration and exploitation by positioning flowers at the extreme of a rectangular workspace that must be scooped by the bees with a propensity that ensures thorough exploration. Exploitation is achieved by a spatial reduction of the best patch subspace over several 3W cycles. Unlike PSO, FBA avoids stagnation and minimizes the possibility of premature convergence that occurs when algorithms are guided by exemplars, honeybees in FBAs are guided by attractors which shift as new flowers are discovered. The proposed algorithm was tested on fifteen standards benchmarks of which three are unimodal while the remaining are complex multimodal spaces with millions of local optima.

The algorithm was compared with two state-of-the-art algorithms PSO and ABC, and the statistically significant result shows that FBA is more efficient than PSO while being competitive with ABC. FBA has been tested extensively in this work, but more experiments need to be done to tune the parameters of FBA for solving more complex test functions at higher dimensions. In addition, adaptive variants of FBA should be developed to reduce the number of parameters that require tuning for high performance. Finally, an information-sharing mechanism will be developed to reduce the overall complexity of the search algorithm.

REFERENCES

Abbass, H. A. (2001). MBO: Marriage in Honey Bees Optimization-A Haplometrosis Polygynous Swarming Approach. *Proceedings of the 2001 IEEE Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546) Vol. 1.*, 207-214.

Alizadehsani, R., Roshanzamir, M., Izadi, N. H., Gravina, R., Kabir, H. D., Nahavandi, D., . . . Fortino, G. (2023). Swarm Intelligence in Internet of Medical Things: A Review. *Sensors, 23*(3), 1466. doi:https://doi.org/10.3390/s2

Altshuler, Y. (2023). Recent Developments in the Theory and Applicability of Swarm Search. *Entropy, 25*(5).

Aslan, S., Karaboga, D., & Badem, H. (2020). A New Artificial Bee Colony Algorithm employing Intelligent Forager Forwarding Strategies. *Applied Soft Computing, 96*.

Belgrade, U. (2015, September 16). *Bee Colony Optimization.* Retrieved September 16, 2015, from http://www.sf.bg.ac.rs/index.php/en-GB/research-fields/814-bee-colony-optimization-bco

Bolaji, A. L., Khader, A. T., Al-Betar, M. A., & Awadallah, M. A. (2013). Artificial Bee Colony Algorithm, Its Variants and Applications. *A Survey, Journal of Theoretical and Applied Information Technology, 47*(2), 434-459.

Chen, X., Tianfield, H., & Du, W. (2021). Bee-foraging Learning Particle Swarm Optimization. *Applied Soft Computing, 102*. doi:10.1016/j.asoc.2021.107134

Cruz, D. P., Maia, R. D., & de Castro, L. N. (2021). A Framework for the Analysis and Synthesis of Swarm Intelligence Algorithms. *Journal of Experimental & Theoretical Artificial Intelligence, 33*, 659-681.

Curkovic, P., & Jerbic, B. (2007). Honey-bees optimization algorithm applied to path planning problem. *International journal of simulation modelling, 6*(3), 154-165.

Dorigo, M., Colorni, A., & Maniezzo, V. (1991). *Positive feedback as a search strategy*. Technical Report 91-016, Politecnico di Milano, Dipartimento di Elettronica, Milan, Italy.

Eberhart, R. C., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence.* Elsevier.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction* (2nd ed.). Pretoria, South Africa: John Wiley & Sons.

Fakhermand, S. M., & Derakhshani, A. (2023). Design Optimization of Soil-Metal Composite Arch Bridges: Recent Swarm Intelligence Applications. *Iranian Journal of Science and Technology, Transactions of Civil Engineering, 47*(1), 373-387.

Gao, W., Liu, S., & Huang, L. (2012). A global best artificial bee colony algorithm for global optimization. *Journal of Computational and Applied Mathematics, 236*(11), 2741-2753.

Haddad, O. B., Afshar, A., & Mariño, M. A. (2006). Honey-Bees Mating Optimization (HBMO) Algorithm. *A New Heuristic Approach for Water Resources Optimization. Water Resources Management, 20*(5), 661-680.

Holland, J. H. (1975). *Adaption in Natural and Artificial System.* . MIT Press.

Janaki, M., & Geethalakshmi, S. N. (2022). A Review of Swarm Intelligence-Based Feature Selection Methods and Its Application. *Soft Computing for Security Applications: Proceedings of ICSCS 2022*, 435-447.

Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization.* Technical report-tr06, Erciyes University, engineering faculty, computer engineering department., Kayseri, Turkiye.

Kaswan, K. S., Dhatterwal, J. S., & Kumar, A. (2023). *Swarm Intelligence: An Approach from Natural to Artificial.* John Wiley & Sons.

Kennedy J. and Eberhart, R. (1995). Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks, 4*, pp. 1942–1948.

Krishnanand , K. N., & Ghose, D. (2005). Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. *in Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, (pp. 84–94). Pasadena, California.

Kumar, A., Chatterjee, J. M., Payal, M., & Rathore, P. S. (2022). Revolutionizing the Internet of Things with Swarm Intelligence. *System Assurances*, 403-436. doi:10.1016/B978-0-323-90240-3.00023-0

Li, X., & Yang, G. (2016). Artificial bee colony algorithm with memory. *Applied Soft Computing, 41*, 362-372.

Mathlouthi , I., & Bouamama, S. (2016). A family of honey-bee optimization algorithms for Max-CSPs. *International Journal of Knowledge-based and Intelligent Engineering Systems, 19*(4), 215-224.

Pan, X. (2016). Genetic-bee Colony Dual-population Self-adaptive Hybrid Algorithm Based on Information Entropy. *Scientific Bulletin of National Mining University, 1*(1), 116.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. (2005). The Bees Algorithm – A Novel Tool for Complex Optimization Problem. In D. T. Pham, E. E. Eldukhri, & A. J. Soroka (Ed.), *Intelligent Production Machines and Systems* (p. 454). Elsevier Science Ltd.

Sato, T., & Hagiwara, M. (1997). Bee System: Finding Solution by a Concentrated Search. *IEEE International Conference on Computational Cybernetics and Simulation* (pp. 3954-395). Orlando, FL, USA: IEEE.

Schumann, A. (. (2020). *Swarm Intelligence: From Social Bacteria to Humans.* CRC Press.

Selvaraj, S., & Choi, E. (2020). Survey of swarm intelligence algorithms. *3rd International Conference on Software Engineering and Information Management*, (pp. 69-73).

Shahzad, M. M., Saeed, Z., Akhtar, A., Munawar, H., Yousaf, M. H., Baloach, N. K., & F, H. (2023). A Review of Swarm Robotics in a NutShell. *Drones, 7*(4), 269.

Solgi, R., & Loáiciga, H. A. (2021). Bee-Inspired Metaheuristics for Global Optimization: A Performance Comparison. *Artificial Intelligence Review, 54*(7), 4967-4996. doi:10.1007/s10462-021-10015-1

Storn, R., & Price, K. V. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.

Teodorovic, D., & Dell'orco, M. (2005). Bee Colony Optimization—A Cooperative Learning Approach to Complex Transportation Problems. *Proceedings of the 16th Mini-EURO Conference on Advanced OR and AI Methods in Transportation*, (pp. 51-60). Poznan. Retrieved September 13-16, 2005

Tzanetos, A., & Dounias, G. (2020). A Comprehensive Survey on the Applications of Swarm Intelligence and Bio-Inspired Evolutionary Strategies. In G. Tsihrintzis, & L. Jain, *Machine Learning Paradigms. Learning and Analytics in Intelligent Systems* (Vol. 18, pp. 337-378). Cham: Springer. doi:https://doi.org/10.1007/978-3-030-49724-8_15

Yang, C., Chen, J., & Tu, X. (2007). Algorithm of Fast Marriage in Honey Bees Optimization and Convergence Analysis. *In Proceedings of the IEEE International Conference on Automation and Logistics* (pp. 1794–1799). Jinan, China: ICAL.