Received: May 27, 2023. Revised: June 17, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Hybrid Artificial Bee Colony and JAYA Algorithm for Linear Antenna Array Synthesis

Al_hussein M. Alturfi[1]* Sonia Goyal[2] Amrit Kaur[2]

[1]Department of Electrical Engineering, University of Misan, Amarah 62001, Iraq
[2] Department of Electronic and Communication Engineering, Punjabi University, Patiala, India
* Corresponding author's Email: al_hussein.m@uomisan.edu.iq

Abstract: Artificial bee colony (ABC) is a very popular algorithm and has been widely used in the literature. The algorithm has some inherent drawbacks, including slow convergence, and poor exploration versus exploitation, among others. In order to deal with these problems, we are hybridizing the ABC algorithm with the JAYA algorithm. The proposed algorithm has the added properties of both ABC and JAYA and has been named as JABC algorithm. The key idea is to add prospective equations of JAYA into ABC for better exploitation, and adding new mutation operators for enhanced exploration and better convergence. The proposed algorithm is tested on CEC 2005 benchmark problems and real world synthesis of linear antenna array (LAA). SLL reduction in five different LAA's is done, for position, amplitude, and phase optimization. We use 10-element, 24-element, 28 element and 40-element LLA for optimization. For performance evaluation, the algorithm is tested with respect to basic ABC, JAYA, spider monkey optimization (SMO), Moth flame optimization (MFO), Chameleon Swarm Algorithm (CSA) and other algorithms. Experimental results on both benchmarks and LAA problems, show that JABC is significantly better as compared to others. Also, statistical analysis using Wilcoxon's test and Friedman tests shows the superior performance of JABC algorithm with respect to other algorithms.

Keywords: Artificial bee colony, Jaya algorithm, CEC benchmark functions, Linear antenna array, Optimization.

1. Introduction

Over the past three decades, many optimization algorithms have been proposed and have been exploited for almost every research problem from medicine to management, business to electronics, operation research, routing problems, and others. The major reason for using these algorithms is due to their simple and linear structure, lesser known parameters for tuning, and better convergence results, among others. These algorithms are commonly known as nature inspired algorithms (NIAs) and are categorized into evolutionary algorithms (EAs) and swarm intelligent algorithms (SIAs).

Among EAs, genetic algorithm (GA) is the oldest known algorithm [1], differential evolution (DE) [2] is another important algorithm. SIAs on the other hand are based on the swarming behaviour of various animal species, and some of the algorithms include naked mole-rat algorithm (NMRA) [3] based on the mating patterns of naked mole-rats, grey wolf optimizer (GWO) [4] based on swarming behaviour of grey wolves found in nature. Other algorithms include sine cosine algorithm (SCA) [5], Harris Hawks Optimization (HHO) [6], Aquila Optimizer [7], guided pelican algorithm (GPA) [8], Stochastic Komodo Algorithm (SKA) [9], Fixed-Step Average and Subtraction-Based Optimizer (FS-ASBO) [10], Attack-Leave algorithm (ALA) [11], quad tournament optimizer (QTO) [12], Multiple Interaction Optimizer (MIO) [13] and others.

For all the major algorithms include initialization, global search (exploration), local search (exploitation), and selection, as their basic operations. Apart from that, the performance of these algorithms is based on various aspects including scaling factor, mutation probability, crossover rate, population, among others. Initial studies showed that parameter

Received: May 27, 2023. Revised: June 17, 2023.

tuning is a crucial step in any algorithm, and can be time-consuming, if based on trial-and-error approach. Premature convergence, local optima stagnation, higher parameter dependence, and other issues make an algorithm prone to poor solutions, and hence can result in reduced performance [14]. This proves that NIAs are prone to various different problems and new algorithms must be designed for solving the problem under consideration.

Artificial bee colony (ABC) is one such algorithm introduced in the recent past [15]. The algorithm is based on the foraging patterns of bees found in nature. The algorithm is based on the division of bees into three groups; employed bees, onlooker bees, and scouts. The bee colony is divided into two parts, where the first half has employed bees, and the second part consists of onlooker bees. For each of these phases, there is only one artificial bee for one food source and employed bee whose food source becomes exhausted, that artificial bee becomes the scout. The algorithm follows many iterations to provide the final best solution. Each search process consists of three steps: i) the employed and onlooker bees move onto the food and nectar amount is calculated, ii) the best scout bee is identified and iii) the scout bee is directed towards the food. Each position of the food source acts as the potential solution of the problem [15].

Many articles have been proposed on the modification and application of the ABC algorithm. Some of these include time varying ABC [16], parallel ABC [17], discrete ABC for cloud services [18], ABC for binary and integer problems [19], ABC for engineering design problems [20], for combinatorial problems [21], among others. Most of the articles discussed above are review articles and provide in-depth details on the variants and applications of ABC. But it has been found in the literature that the algorithm still suffers from various problems, including local optima stagnation, poor exploration, and slow convergence, among others [22]. Also, very little work has been done to improve the parameters of the algorithm, and simultaneous efforts need to be done. Most of the work done on ABC is concentrated around equation modifications or simple applications to the problem, and less effort has been made to make it a generic problem solver. In order to deal with these problems, JAYA algorithm [23] based modification is added to ABC. The JAYA algorithm is simple in structure and is based on the concepts of moving the solution obtained for the problem towards the best solution and simultaneously avoiding the worst solution. The algorithm is found to be highly reliable due to its parameter less nature and hence is found to provide viable solutions. But as the problem complexity increases, modifications must be added to make it suitable for challenging problems.

The above said limitations of existing algorithms have motivated the authors to propose a new algorithm. The new algorithm has been named as JABC and is meant to overcome the problems of ABC as well as JAYA. The modifications are added by enhancing the equations of ABC for both employed and onlooker phase, and for scout phase JAYA based equations are used. Modification are added in the generalized parameters (including simulated annealing based parametric adjustments) of the employed and onlooker phase. This helps the algorithm performing extensive exploration and exploitation of the search space, along with a balanced operation. Introducing JAYA into the scout phase makes the scout phase parameter independent and also helps to provide better convergence properties. More details about the proposed modifications are presented in consecutive subsections. For performance evaluation, we use CEC 2005 benchmark problems [24], consisting of unimodal, multimodal, and some fixed dimension test problems. These test functions are highly challenging and are used for performance evaluation of most of the newly proposed algorithms. Apart from the benchmark problems, synthesis of linear antenna array is also done using the proposed JABC algorithm.

Antennas are the backbone of modern wireless devices, including mobiles, radios, radars, satellites, among others. In these technologies, set of antennas are used instead of a single antenna, and this is because of their capabilities in controlling the main lobe, radiation pattern and adjust parameters such as positions, excitation phases, and excitation currents. Antenna arrays, in general, help in reducing the power consumption, side lobe level, and enhance signal-to-noise ratio; and can have different geometries, namely rectangular, elliptic, linear, hexagonal and circular. Among all, linear antenna arrays (LAAs) are the most common and the simplest among all arrays. In a LAA, the elements are placed along a single axis, owing to steer in a particular direction, and provide an omnidirectional radiation pattern for diversity in one plane. Synthesis of LAA is a very intuitive subject and has been exploited vastly in the literature. Many optimization algorithms have been applied for LAA synthesis, including Grasshopper Optimization Algorithm (GOA) [25], Moth flame optimization (MFO) [26], Mayfly Algorithm (MA) [27], Symbiotic Organism Search (SOS) [28], Modified Seagull Optimization Algorithm (MSOA) [29], Particle Swarm

Table 1. List of variables and notations used in equations

Variables	Notations
$x_{i,k}$	Is i^{th} solution in k^{th} dimension
$x_{min,k}$	Lower bound of the problem
$x_{max,k}$	Upper bound of the problem
v_i^t	New solution in the tth iteration
x_i^k	solution in k^{th} dimension
x_j^t	Random solution
φ	Mutation operator
fit_i	Fitness function
$f_i(x_i)$	Fitness of x_i
p_m	Probability
x_{best}	Best solution
x_{worst}	Worst solution
r_1, r_2	Random numbers
N	Population size
Dim	Dimension size
AF	Array factor
I_n	Excitation amplitude
α_n	Excitation phase
d_i	Inter-element spacing
K	Wave number
x_n	Element position

Optimization (PSO) [30], Chameleon Swarm Algorithm (CSA) [31], Cuckoo Search (CS) [32], Enhanced Firefly Algorithm (EFA) [33], Bat Flower Pollinator (BFP) [34], Invasive Weed Optimizer (IWO) [35], Cat swarm Optimization (CSO) [36], and others [37].

For comparative study with respect to CEC benchmarks, ABC, BBO, FA and other algorithms have been used. A statistical analysis using Wilcoxon's ranksum and Friedman test [38] is also done to prove the significance of the proposed algorithm.

Table 1 provides a brief overview of the variables and notations used in the equations.

Overall, the paper is divided into four sections. Apart from the introduction in the first section, the proposed algorithm, along with why and how the algorithm is proposed, is presented in the second section. The third section is the result and discussion section. This section is divided into two parts, where in the first subsection we provide benchmark results and in the second subsection, the linear antenna array synthesis problem is addressed. The final section provided details about conclusions and some future prospects.

2. The proposed JABC algorithm

The ABC algorithm is one among the most effective algorithms and this can be better understood from the fact that the basic ABC paper has received more than 8800 citations [15]. The suitability of the algorithm for large scale problems and high dimensions is still a matter of concern, and it suffers from poor exploration, poor exploitation, and has degraded convergence patterns [22]. Many papers have been proposed to improve its performance and for application to specific problems [14].

In this section, we provide details about the proposed JABC algorithm. Since JABC uses the basic structure of both ABC and JAYA, no explicit details on the basics of ABC and JAYA are presented in this paper. For implementation details of both of these algorithms, the readers can refer to [15, 23]. The aim of the proposed JABC algorithm is to mitigate poor exploitation and exploration, slow convergence speed, and unbalanced local and global search.

Initialization: The first step deals with initialization of N food sources in the random manner for the dimension d of the problem. This phase is mathematically given as:

$$x_{i,k} = x_{min,k} + r(0,1) \times (x_{min,k} - x_{max,k}) \quad (1)$$

where, $i \in [1, 2, \ldots.. n], k \in [1, 2, \ldots d]$, $x_{i,k}$ represents , i^{th} solution in k^{th} dimension; r(0, 1) is a random number; $x_{min,k}$ and $x_{max,k}$ represents the lower bound and the upper bounds of the problem. This phase is the same for the whole algorithm and provides the initial set of solutions for the performance evaluation.

Employed bee phase: The second phase of the proposed algorithm is the employed phase and is similar to the basic ABC algorithm with added modifications. The phase consists of employed bees that searches for food sources (v_i^k) with more amount of nectar among the neighbouring food sources (x_i^k). The generalized equation for this phase is given by:

$$v_i^t = x_i^t + \varphi(x_i^t - x_j^t) \quad (2)$$

Where x_j^t is a random food source in the i^{th} direction, and φ is a random number generated in the range of [0, 1]. In the present case, a simulated annealing based mutation operator is used to formulate new values of φ . The mathematical formulation for simulated annealing based φ is given by :

$$\Phi = \gamma_{min} + (\gamma_{max} - \gamma_{min}) \times r^{k-1} \quad (3)$$

Where γ_{max} = 0.95, γ_{min}= 0.45, &k = rand[0,1] are the parameters of simulated annealing mutation operator. this mutation operator helps to provide better exploration operation, and helps in improved convergence patterns of the proposed algorithm [39]. After generating a new food source v_i, its fitness is compared with respect to x_i. The fitness fit_i for the solution x_i corresponding to the $f_i\,(x_i)$ objective function is given by:

$$fit_i = \begin{cases} \frac{1}{1+f_i(x_i)} & if\ f_i(x_i) \geq 0 \\ 1+|f_i(x_i)| & if\ f_i(x_i) < 0 \end{cases} \quad (4)$$

Onlooker bee phase: This phase is governed by unemployed bees. These unemployed bees take the information of food sources from the employed bees and choose the best food source for collecting nectar. Each food source is selected based on a certain probability p_m and this probability is chosen by using

$$p_m = \frac{fit_i}{\sum fit_i} \quad (5)$$

After choosing a food source x_i, new neighbouring solutions are found by Eq. (2) and its fitness value is evaluated using greedy selection.

Greedy selection: The last phase of the algorithm is greedy selection, and in this phase, we find the best solution. The phase generally compares the current best solution with the previous best and choses the best among the two. This best solution is then updated over the course of iterations and after a certain set of iteration or until the stopping criteria is achieved, the final best solution is retained. The generalized equation for this phase is given by:

$$v_i^{t+1} = \begin{cases} v_i^t & if\ f(v_i) < f(x_i^t) \\ x_i^t & otherwise \end{cases} \quad (6)$$

Where, v_i^{t+1} is the current iterative best solution, x_i^t is the previous best iterative solution and $f(x_i^t)$ is the fitness corresponding to x_i^t solution.

Scout bee phase: Those unemployed bees who randomly select food sources are scouts. This phase is activated if the solution quality does not improve after a certain number of trials. If x_i is abandoned, the new solution becomes the employed bee and is generated by using JAYA algorithm [23]. The major reason for the use of JAYA algorithm in this phase is the parameter less nature of this algorithm. Apart from that, due to the movement of new solutions toward the best and away from the worst solution, the added modification helps the algorithm in local optima avoidance problem. The generalized equation for this phase is given by

$$x_i^{t+1} = x_i^t + r_1(x_{best} - x_i^t) - r_2(x_{worst} - x_i^t) \quad (7)$$

where r_1 and r_2 are random numbers, x_{best} and x_{worst} are the best and the worst solutions, $(x_{best} - x_i^t)$ indicates the capability of each solution to move towards the best solution and $(x_{worst} - x_i^t)$ represents the capability to move away from the worst solution. The overall movement of these individuals helps the algorithm in local optima avoidance and hence provides better convergence. The pseudocode of the JABC algorithm is given in Algorithm 1.

Algorithm 1 Pseudocode of JABC algorithm

```
1: Begin
2: Define: the size of the population ($N$)
3: stopping criteria; problem dimension ($Dim$)
4: i= 1: maximum number of iterations
5: Employed bee phase using Eq. (2)
6: Onlooker bee phase using Eq. (2) & (5)
7: Evaluate fitness and perform greedy selection
   using Eq. (6)
8: Scout phase using Eq. (7)
9: update $\varphi$ using Eq. (3)
10: close;
11: update final best
12: End
```

3. Result and discussion

This section deals with the analysis of the proposed JABC with respect to benchmark and synthesis of LAAs. The whole section is divided into two subsections, where the first subsection deals with the performance evaluation of the proposed algorithm for the CEC 2005 benchmark test suite [24] and the second subsection provides extensive results on the synthesis of LAA problem. All the simulations are performed on a Windows 10 64-bit operating system with 8GB RAM, Intel Core i7 processor, and MATLAB 2022a.

3.1 Performance evaluation for CEC 2005 test functions in comparison with other algorithms

The performance of JABC is tested on eight benchmark problems and a comparison with respect to ABC [16], bat algorithm (BA), firefly algorithm (FA), and BBO are taken from [34]. These algorithms are competitive and have been found to provide viable solutions for the problems under test. The parameter corresponding to each of the algorithm is

Table 2. Parameter settings of various algorithms

Algorithm	Parameters	Values	Algorithm	Parameters	Values
FA	Number of fireflies	20	BBO	Population Size	20
	Alpha (α)	0.25		Mutation probability	0.25
	Beta (β)	0.20		Habitat modification probability	1
	Gamma (γ)	1		Maximum number of iterations	500
	Maximum number of iterations	500		Stopping Criteria	Max. Iteration.
	Stopping Criteria	Max Iteration.			
BA	Population size	20	JABC	Colony size (SN)	20
	Loudness	0.5		Number of food sources	SN/2
	Pulse rate	0.5		Limit	100
	[Qmin, Qmax]	[0,1]		Maximum number of iterations	500
	Maximum number iterations	1000		Stopping Criteria	Max Iteration.
	Stopping Criteria	Max Iteration.			
ABC	Colony size (SN)	20			
	Number of food sources	SN/2			
	Limit	100			
	Maximum number iterations	500			
	Stopping Criteria	Max Iteration.			

Table 3. Benchmark functions used in the simulation

Problems	Objective Function	Range	Optimum Value	D
$f_1(x)$	$-\sum_{i=1}^{4} \alpha_i \exp[-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2]$	[0, 1]	−3.86278	3
$f_2(x)$	$-\sum_{i=1}^{4} \alpha_i \exp[-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2]$	[0, 1]	−3.32237	6
$f_3(x)$	$-\sum_{j=1}^{5} [\sum_{i=1}^{4} ((x_i - C_{ij})^2 + \beta_j)^{-1}]$	[0, 10]	−10.1532	4
$f_4(x)$	$-\sum_{j=1}^{7} [\sum_{i=1}^{4} ((x_i - C_{ij})^2 + \beta_j)^{-1}]$	[0, 10]	−10.4029	4
$f_5(x)$	$-\sum_{j=1}^{10} [\sum_{i=1}^{4} ((x_i - C_{ij})^2 + \beta_j)^{-1}]$	[0, 10]	−10.5364	4
$f_6(x)$	$10D + \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i)]$	[−5.12, 5.12]	0	30
$f_7(x)$	$\left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	[−5, 5]	−1.0316	2
$f_8(x)$	$(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3{x_1}^2 - 14x_2 + 6x_1x_2 + 3{x_2}^2))(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12{x_1}^2 + 48x_2 - 36x_1x_2 + 27{x_2}^2))$	[−2, 2]	3	2

given in Table 2 and the description of the benchmarks is given in Table 3. The results for comparison are presented in Tables 4 and 5 gives the statistical results for each of the algorithms under comparison.

Experimental results: From the results in Table 4, best values are shown in the bold text. For functions, f_2, f_5, f_6, f_7 and f_8 the standard deviation of JABC is much better except for f_1 in which FA is better, f_3 where ABC is better and f_4 where BA is better. Mean for seven function is better except for only f_2 and f_3 where FA is better. As far as best value is concerned, JABC gives best for most of the test functions except for f_6 where BA is better. The results show that JABC algorithm performs better than ABC, BBO, BA and FA for most of the test functions. The proposed algorithm is also able to achieve better mean and standard deviation values than competing algorithms.

Statistical Testing: Wilcoxon's rank-sum test and Friedman rank (f-rank) [38], are used to validate

Table 4. Simulation results

Function	Algorithm	Best	Worst	Mean	Standard Deviation
$f_1(x)$	ABC	−3.77541	−2.41101	−3.23971	4.151E−01
	BBO	−3.22341	−0.00242	−0.96732	9.551E−01
	BA	**−3.86282**	−3.08983	-3.78554	0.23793
	FA	**−3.86283**	**−3.86284**	**-3.86284**	**3.312E-007**
	JABC	**−3.86284**	−3.79512	−3.85485	1.682E−02
$f_2(x)$	ABC	−2.19631	−0.72901	−1.38161	4.643E−01
	BBO	−3.14523	−1.90592	−2.75012	3.023E−01
	BA	**−3.32243**	**−3.20313**	−3.26273	0.06121
	FA	**−3.32221**	−3.19153	**−3.26724**	0.06262
	JABC	**−3.32242**	−3.09414	−3.25145	**8.673E−03**
$f_3(x)$	ABC	−10.10731	−2.59283	−6.61141	**3.09561**
	BBO	**−10.15322**	−2.63042	−6.14442	3.47912
	BA	−10.15251	**−2.63051**	−5.01863	3.18792
	FA	−10.15281	−2.63042	−6.78843	3.81552
	JABC	**−10.15322**	**−2.63054**	**−7.90952**	3.51641
$f_4(x)$	ABC	−10.50543	−1.66804	−5.90551	3.22573
	BBO	−10.40284	**−2.76595**	−7.60972	3.54632
	BA	−10.40293	−1.83763	−4.02643	**2.49083**
	FA	−10.40282	−2.75192	−9.25424	2.80251
	JABC	**−10.40291**	−2.75191	**−9.25673**	2.79952
$f_5(x)$	ABC	−10.46422	−1.85082	−5.35521	3.42953
	BBO	−10.53631	−2.80663	−7.32432	3.65851
	BA	**−10.53642**	−1.67662	−4.19373	3.30052
	FA	−10.53623	−10.53471	−10.53554	4.85E-004
	JABC	**−10.53644**	**−10.53642**	**−10.53645**	**7.292E−06**
$f_6(x)$	ABC	4.22E+01	9.20E+01	6.76E+01	1.371E+01
	BBO	9.67411	2.20E+01	1.95E+01	3.104249
	BA	**8.10E-09**	12.92344	4.07932	3.194044
	FA	3.63E-06	1.10E-04	4.06E-05	3.222E-05
	JABC	1.31E-08	**2.49E-07**	**1.01E-07**	**8.553E-08**
$f_7(x)$	ABC	**−1.03161**	−1.02611	−1.03052	1.501E−03
	BBO	−1.02342	−0.04792	−0.73143	3.452E−01
	BA	**−1.03163**	−0.21553	−0.78683	0.38372
	FA	**−1.03163**	**−1.03162**	**−1.03163**	1.221£-006
	JABC	**−1.03161**	**−1.03162**	**−1.03161**	**4.582E−09**
$f_8(x)$	ABC	3.00032	3.09043	3.01902	2.523E−02
	BBO	**3.00002**	**3.00003**	**3.00003**	**0.00E+00**
	BA	**3.00003**	84.00004	16.50002	25.53942
	FA	**3.00004**	**3.00005**	**3.00001**	1.482E-05
	JABC	**3.00000**	**3.00000**	**3.00000**	2.442E−08

the applicability of the proposed JABC statistically. Wilcoxon's ranksum test is done to provide details of results in terms of *win(w)*, *loss(l)* and *tie(t)*. Here *w* given as " + " means that the proposed algorithm is better than the algorithm under comparison, *l* given by " − " means the proposed algorithm does not provide better results than the test algorithm, and *t* given by " = " stands for equality in results. The results in Table 5 shows that for most of the cases, our proposed JABC is better and significant with respect to others.

3.2 Synthesis of linear antenna arrays

This section provides extensive results of the proposed algorithm for the synthesis of LAA. A total of five examples have been used and are meant for optimization of phase, amplitude and positions of array elements.

3.2.1. Problem formulation

The wider applicability of LAA due to its easier implementation and simplicity, has been widely explored in the literature. Fig. 1 shows a LAA with $2N$ number of elements distributed along the axis without any element at the origin.

The array factor for the geometry with odd and even number of elements is given by Eqs. (8) and (9), respectively [25].

Table 5. Wilcoxon's ranksum and Freidman test results

Test function		Algorithm				
		ABC	BBO	BA	FA	JABC
$f_1(x)$	p-rank	-	-	-	+	N/A
	f-rank	4	5	3	1	2
$f_2(x)$	p-rank	-	-	-	-	N/A
	f-rank	5	4	3	2	1
$f_3(x)$	p-rank	-	-	-	-	N/A
	f-rank	5	2	4	3	1
$f_4(x)$	p-rank	-	-	+	-	N/A
	f-rank	5	4	1	3	2
$f_5(x)$	p-rank	-	-	-	-	N/A
	f-rank	4	5	3	2	1
$f_6(x)$	p-rank	-	-	-	-	N/A
	f-rank	5	4	3	2	1
$f_7(x)$	p-rank	-	-	-	-	N/A
	f-rank	3	4	5	2	1
$f_8(x)$	p-rank	-	+	-	-	N/A
	f-rank	4	1	5	3	2
w/l/t		0/8/0	1/7/0	1/7/0	1/7/0	
overall f-rank		35	29	27	18	10
Average f-rank		5	4	3	2	1

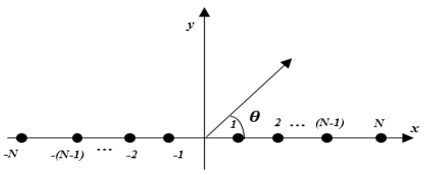

Figure. 1 Geometry of a systematic LAA

$$AF\ (\theta) = I_o\, exp(j\theta_0) + 2\sum_{n=-N,n\neq 0}^{N} I_n\, exp\ (j\ [kx_n cos\ (\theta) + \alpha_n] \qquad (8)$$

$$AF\ (\theta) = 2\sum_{n=-N,n\neq 0}^{N} I_n\, exp\ (j\ [kx_n cos\ (\theta) + \alpha_n] \qquad (9)$$

Where α_n and I_n are the excitation phase and amplitude respectively for the n^{th} element feeding current, $x_n = \sum_{i=1}^{n} d_i$ is the position and d_i is inter-element spacing, having k = 2π/λ as the wave number. After solving the above equations for $\alpha_n = \alpha_{-n}$, $x_n = x_{-n}$ & $I_n = I_{-n}$, the array factor becomes:

$$AF\ (\theta) = I_o \exp(j\theta_0) + 2\sum_{n=1}^{N} I_n\, cos\ (j\ [kx_n cos\ (\theta) + \alpha_n] \qquad (10)$$

$$AF\ (\theta) = 2\sum_{n=1}^{N} I_n\, cos\ (j\ [kx_n cos\ (\theta) + \alpha_n] \ (11)$$

The array variables (α_n, I_n, and x_n) are optimized in present case, by suppressing the side lobe level (SLL). The fitness function for minimization is given by [25]:

$$\text{Fit function} = min\left[max\ \left(20\,log\frac{|AF(\Phi)|}{max|AF(\theta)|}\right)\right] \ (12)$$

Where [0, Φ] is the side lobe region, and it depends on the element number.

3.2.2. Optimization of element amplitude (I$_n$)

The array factor for this case is given for even and odd elements, respectively by:

$$\text{AF}\ (\theta) = 2\sum_{\text{n}=1}^{\text{N}} I_n\, cos\ [(n - 0.5)\pi cos\ (\theta)]\ (13)$$

$$\text{AF}\ (\theta) = I_o + 2\sum_{\text{n}=1}^{\text{N}} I_n\, cos\ [\text{n}\pi cos\ (\theta)]\ (14)$$

In this case we have used 10-element and 24-element LAA. A comparison with some of the recent algorithms is performed.

3.2.2.1. Example 1: 10-element LAA

For a 10-element LAA, the proposed JABC is compared with ABC, JAYA, GOA [25], MFO [26], MA [27], SOS [28], MSOA [29], PSO [30], and CS [31] algorithms. The best amplitudes for these techniques are given in Table 6. Figs. 2 and 3 show the radiation patterns and convergence curves, respectively. The results show that among all the algorithms, JABC performed the best and it also showed the superior performance of JABC.

3.2.2.2. Example 2: 24 Elements LAA

For a population size of 50, and iteration set of 150, in the current case, we optimize the amplitude of 24 element arrays. The maximum obtained SLL obtained by JABC is −49.09 dB which is comparatively better than SOS, Taguchi and others. Here Table 7 shows the optimum amplitude, and Figs. 4 and 5 show the radiation pattern and convergence curves for 24-element arrays, respectively.

3.2.3. Optimization of element positions (x_n)

In this section, by selecting the optimum positions for each LAA element, the minimum peak SLL is achieved. In order to achieve this, the amplitudes and phases should be fixed, i.e. ($a_n = 0$ & $I_n = 1$), so the array factor becomes:

$$AF(\theta)= 2\sum_{n=1}^{N} cos\,[kx_n cos\,(\theta)\,] \tag{15}$$

The appropriative position of elements is important. Because mutual coupling effects can occur if antennas are placed too closely together, grating lobes result if placed too far away. Thus, to overcome the drawbacks indicated, the following conditions [31] must be achieved:

$$|x_i - x_j| > 0.25 \tag{16}$$

$$\text{minimum } \{x_i\} > 0.125\lambda \quad i=1,2,3,..,N. \; i \neq j \tag{17}$$

3.2.3.1. Example 3: 10 Elements LAA

In this example, the element positions are optimized, and the proposed algorithm JABC is compared with respect to CSO [36], GOA [25], PSO [36], SMO [40], ABC, JAYA and conventional uniform antenna array. From the results in Table 8, we can see that JABC is far better when compared to GOA, PSO, SMO and uniform antenna array and is comparative with respect to ABC, JAYA and CSO. The radiation patterns and convergence curve for this case is given in Figs. 6 and 7, respectively.

3.2.3.2. Example 4: 28 elements LAA

For a 28 element LAA, the optimization of element positions is done in this example. Table 9 presents the results of all the algorithms, namely JAYA, CSO, PSO, ABC, and uniformly distributed LAA. The results show that for an FNBW at 10.8^o, the maximum SLL is achieved by JABC and equals −25.8dB, which is comparatively better compared to ABC (−24.06dB), JAYA (−24.61dB), CSO (−24.53dB), and others. The radiation pattern and convergence curves for 28-element LAA is given in Figs. 8 and 9, respectively.

3.2.4. Optimization of elements phases (α_n)

As a uniform array, we set $I_n = 1$ and the spaces between elements (d= λ/2). Initial phase values are uniformly distributed in (0, 180). Elements Phases are considered to be symmetric as ($\alpha_n = \alpha_{-n}$ n = 1, 2, . . ., *N)***.** Where (α_n) phase of the n^{th} element. So, the AF becomes as the following:

$$AF(\theta)=2\sum_{n=1}^{N} exp(j\alpha_n)cos[(n-0.5)\,\pi cos(\theta)] \tag{18}$$

3.2.4.1. Example 5: 40 elements LAA

From Table 10, the maximum SLL using JABC is −18.18 dB which is comparatively close when compared to SOS has an SLL of −18.02 dB. But, the radiation patterns and convergence curves in Fig. 10& 11 prove the significance of JABC over other algorithms under comparison.

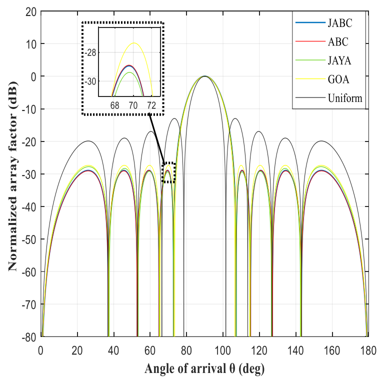

Figure. 2 Radiation patterns of 10-element

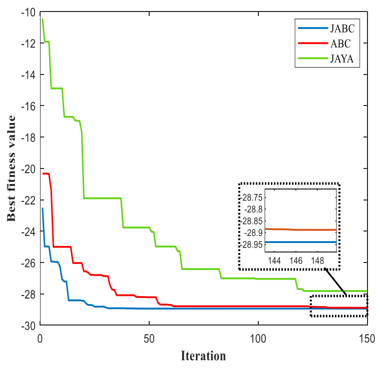

Figure. 3 Convergence curves of 10-element

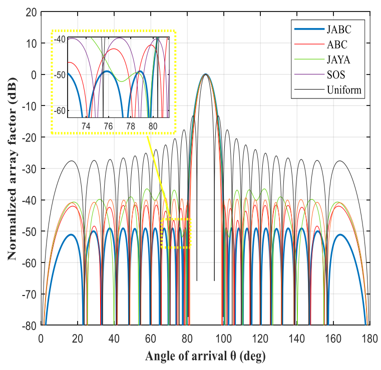

Figure. 4 Radiation patterns of 24-element

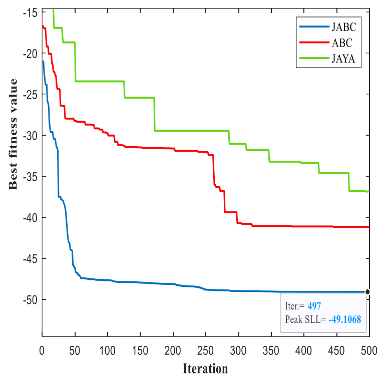
Figure. 5 Convergence curves of 24-element

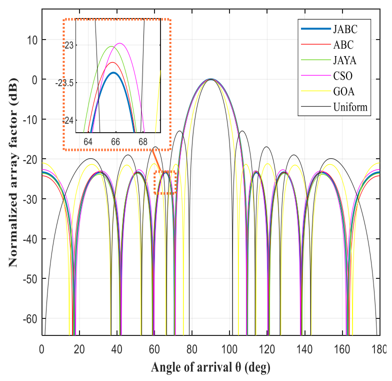
Figure. 6 Radiation patterns of 10-element

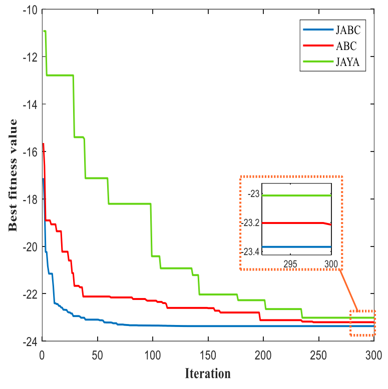
Figure. 7 Convergence curves of 10-element

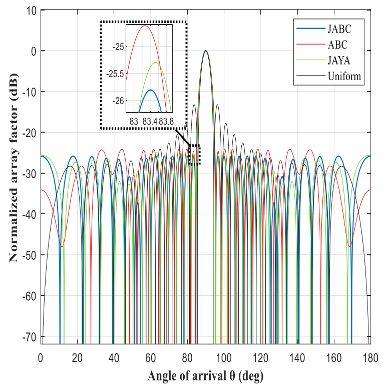
Figure. 8 Radiation patterns of 28-element

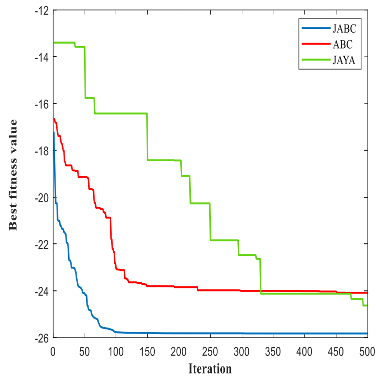
Figure. 9 Convergence curves of 28-element

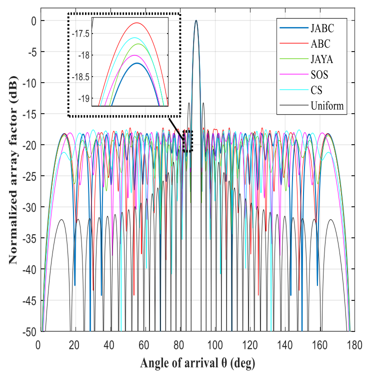
Figure. 10 Radiation patterns of 40-element

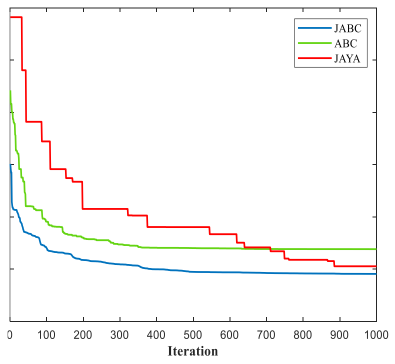
Figure. 11 Convergence curves of 40-element

Table 6. Optimized amplitude for a 10-element LAA obtained with suggested algorithms compared to other techniques

Evolutionary algorithm	Optimized element amplitudes $I_1, I_2, \ldots, I_5$	Peak SLL (dB)	FNBW
JABC	1.0000, 0.8824, 0.6798, 0.4449, 0.2808	-28.94	34.42◦
ABC	1.0000, 0.8817, 0.6797, 0.4455, 0.2793	-28.89	34.42◦
JAYA	1.0000, 0.8836, 0.6822, 0.4369, 0.2894	-27.78	34.42◦
GOA [25]	1.0000, 0.8892, 0.6962, 0.4684, 0.3208	-27.36	33.08◦
MFO [26]	1.0000, 0.8962, 0.6966, 0.4935, 0.2965	-26.07	32.59◦
MA [27]	1.0000, 0.8922, 0.7036, 0.4791, 0.3400	-26.70	32.59◦
SOS [28]	1.0000, 0.8985, 0.7189, 0.5017, 0.3856	-25.28	31.40◦
MSOA [29]	1.0000, 0.8887, 0.6944, 0.4657, 0.3154	-27.52	33.59◦
PSO [30]	1.0000, 0.9010, 0.7255, 0.5120, 0.4088	-24.62	30.80◦
CSA [31]	0.9992, 0.8901, 0.6996, 0.4738, 0.3312	-26.99	32.59◦
CS [32]	1.0000, 0.9019, 0.7273, 0.5153, 0.4157	-24.43	30.80◦
Uniform	1.0000, 1.0000, 1.0000, 1.0000, 1.0000	-12.97	23.00◦

Table 7. Optimized amplitude for a 24-element LAA obtained with suggested algorithms compared to other techniques

Evolutionary algorithm	Optimized element amplitudes $I_1, I_2, \ldots, I_{12}$	Peak SLL (dB)	FNBW
JABC	1.0000, 0.9642, 0.8958, 0.8011, 0.6879, 0.5655,0.4439, 0.3288, 0.2302, 0.1468, 0.0865, 0.0502	-49.09	20.55◦
ABC	1.0000, 0.9718, 0.9061, 0.8287, 0.7193, 0.6180, 0.4981, 0.3911, 0.2851, 0.1884, 0.1240, 0.0932	-41.66	18.55◦
JAYA	1.0000, 0.9429, 0.8744, 0.8112, 0.7299, 0.5544, 0.4148, 0.3445, 0.2173, 0.1673, 0.0991, 0.0590	-36.53	20.55◦
SOS [28]	1.0000, 0.9699, 0.9143, 0.8387, 0.7420, 0.6368,0.5273, 0.4145, 0.3149, 0.2243, 0.1515, 0.1236	-39.37	17.54◦
CSA [31]	0.9968, 0.9737, 0.9062, 0.8395, 0.7276, 0.6332, 0.5093, 0.4048, 0.3031, 0.2076, 0.1449, 0.1052	-40.90	17.54◦
EFA [32]	1,0000, 0.9990, 0.9387, 0.8590, 0.7672, 0.6443, 0.5510, 0.4515, 0.3391, 0.2495, 0.1582, 0.1443	-37.36	16.54
PSO [30]	1.0000, 0.9712, 0.9226, 0.8591, 0.7812, 0.6807,0.5751, 0.4768, 0.3793, 0.2878, 0.2020, 0.2167	-34.46	15.54◦
Uniform	1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000,1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000	-13.18	9.53◦

Table 8. Optimized positions for a 10-element LAA obtained with suggested algorithms compared to other techniques

Evolutionary algorithm	Optimized element positions	Peak SLL (dB)	FNBW
JABC	0.1492λ, 0.3992λ, 0.7817λ, 1.0747λ, 1.6599λ	-23.36	38.6◦
ABC	0.1367λ,0.4076λ, 0.7759λ, 1.0749λ, 1.6548λ	-23.22	38.6◦
JAYA	0.1478λ, 0.3985λ, 0.7830λ, 1.0681λ, 1.6514λ	-23.01	38.6◦
CSO [36]	0.1510λ, 0.4110λ, 0.7890λ, 1.1040λ, 1.6840λ	-22.89	37.8◦
MA [27]	0.2915λ, 0.5567λ, 0.9456λ, 1.2654λ, 1.8722λ	-22.79	30.6◦
GOA [25]	0.3360λ, 0.4190λ, 1.0120λ, 1.4160λ, 2.1000λ	-21.31	29.5◦
PSO [36]	0.2600λ, 0.5100λ, 1.0180λ, 1.4690λ, 2.1400λ	-20.72	28.5◦
SMO [40]	0.2360λ, 0.5280λ, 1.0070λ, 1.4710λ, 2.1260λ	-20.25	28.5◦
Uniform	0.2500λ, 0.7500λ, 1.2500λ, 1.7500λ, 2.2500λ	-12.96	23◦

Table 9. Optimized positions for a 28-element LAA obtained with suggested algorithms compared to other techniques

Evolutionary algorithm	Optimized element positions	Peak SLL	FNBW
JABC	0.2339λ, 0.4994λ, 0.8938λ, 1.2786λ, 1.5790λ, 2.0768λ, 2.3728λ, 2.8697λ, 3.2608λ, 3.8343λ, 4.3208λ, 4.9977λ, 5.8325λ, 6.6690λ	-25.80	10.8◦
ABC	0.2390λ, 0.4890λ, 0.9119λ, 1.1619λ, 1.6726λ, 1.9547λ, 2.3402λ, 2.7994λ, 3.2090λ, 3.7500λ, 4.2910λ, 4.8917λ, 5.6927λ, 6.4138λ	-24.06	10.9◦
JAYA	0.2330λ, 0.5084λ, 0.9095λ, 1.3151λ, 1.6198λ, 2.1014λ, 2.4002λ, 2.9195λ, 3.3246λ, 3.8986λ, 4.4290λ, 5.0937λ, 5.9251λ, 6.7293λ	-24.61	10.6◦
CSO [36]	0.2344λ, 0.5280λ, 0.9224λ, 1.2965λ, 1.6549λ, 2.1427λ, 2.4387λ, 2.9369λ, 3.3753λ, 3.9280λ, 4.4091λ, 5.1167λ, 5.9188λ, 6.7422λ	-24.53	10.5◦
PSO [36]	0.1703λ, 0.6430λ, 0.9509λ, 1.4245λ, 1.7849λ, 2.0397λ, 2.4511λ, 3.0522λ, 3.0522λ, 3.6249λ, 4.0476λ, 4.6302λ, 5.2984λ, 6.7118λ	-21.89	10◦
Uniform	0.2500λ, 0.7500λ, 1.2500λ, 1.7500λ, 2.2500λ, 2.7500λ, 3.2500λ, 3.7500λ, 4.2500λ, 4.7500λ, 5.2500λ, 5.7500λ, 6.2500λ, 6.7500λ	-13.27	8◦

3 Conclusion

Nature inspired algorithms have dominated optimization research, and various new algorithms have been proposed in the recent past to solve various challenging problems. ABC is one such algorithm which has been used to solve various problems and still suffers from the problems of local optima stagnation, parameter optimization, among others. To overcome these problems, a novel hybrid variant of ABC clubbed with JAYA, namely JABC algorithm, is proposed. The proposed algorithm has the added advantages of both ABC and JAYA, and is done by incorporating JAYA into the scout phase of ABC.

Table 10. Optimized phases for a 40-element linear antenna array (LAA)

Evolutionary algorithm	Optimized element phases $\alpha_1, \alpha_2, \ldots\ldots, \alpha_{20}$(deg.)	Peak SLL in dB	FNBW
JABC	73.485, 71.919, 74.404, 69.146, 66.585, 69.589, 71.782, 61.804, 64.706, 55.43, 65.562, 86.243, 6.4126, 45.037, 96.356, 177.48, 49.21, 109.66, 68.351, 73.276	-18.18	6.2◦
ABC	122.51, 106.14, 106.93, 103.23, 117.01, 109.05, 106.66, 122.18, 113.8, 126.04, 129.77, 128.18, 154.54, 65.856, 140.34, 35.447, 50.567, 128.97, 130.64, 83.731	-17.25	6.5◦
JAYA	90.549, 92.692, 87.634, 89.066, 92.932, 74.032, 87.602, 75.37, 64.679, 106.8, 72.997, 77.751, 112.9, 0.6161, 94.598, 115.31, 176.24, 66.407, 88.068, 89.272	-17.74	6.3◦
SOS [28]	28.3636, 25.0046, 22.2290, 31.1901, 23.7626, 17.3337,15.5147, 39.0199, 18.1678, 7.8822, 1.8298, 60.0022, 0, 0.0146, 0.0161, 148.3908, 45.0096, 56.1693, 61.9867, 2.1350	-18.02	6.6◦
CS [32]	45.9692, 39.7155, 39.6464, 36.8069, 41.0828, 42.4519, 50.2623, 32.5464,36.8147, 34.4894, 30.8162, 16.1212, 81.8888, 20.4923, 41.963, 177.6511,30.7085, 53.6503, 35.3756, 87.3351	-17.59	6.4◦
BBO [37]	90.4185, 90.5331, 97.2825, 90.2466, 88.3840, 97.1507, 90.0002, 90.3497, 97.2596, 85.9950, 75.0002, 115.5026, 71.8604, 0.3610, 122.9166, 97.0247, 178.8087, 83.3081, 83.9670, 79.2057	-17.96	6.2◦
Uniform	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	-13.24	6.2◦

The enhanced scout phase helps the algorithm in providing better convergence speed and hence avoids local optima stagnation. Apart from that, the addition of simulated annealing based mutation weight provides better exploration and keeps a balanced exploration as well as exploitation operation. The simulation results are evaluated using CEC 2005 benchmark problems and synthesis of LAA. A total of five examples have been used for SLL reduction, for element, amplitude and phase optimization. Two statistical tests namely Wilxocon's test and Freidman test prove the significance of the algorithm statistically.

For optimizing the element amplitudes, 10-element, and 24-element LAA's are used. We find that a maximum SLL of -28.94 dB at FNBW 34.42^o for 10-element and -49.09 dB at FNBW 20.55^o is achieved using 24-element LAA respectively, and is significantly better than MSOA, CSA, PSO, and others. To optimize positions, 10-element, and 28-element LAA's are used. Here we find that a maximum SLL of -23.36 dB at FNBW 38.6^o for 10-element, and -25.80 dB at FNBW 10.8^o for 28-element LAA is achieved. The final case uses phase optimization of a 40-element LAA where the peak SLL achieved is -18.18 dB at FNBW 6.2^o. Overall, in all the cases, the proposed JABC performs significantly better as compared to other algorithms such as MA, CS, CSO, PSO, ABC, JAYA, among others.

For future, JABC algorithm can be further enhanced by analyzing the impact of all of its parameters. New mutation operators and inertia weight operators can be highlighted. The proposed algorithm can be applied to various optimization problems such as task scheduling, parametric estimation, big data applications, image segmentation, cloud computing, classification and other design problems. The algorithmic enhancements can be added by using new and prospective equation based modifications, population adaptation and other prospective changes for performance enhancement.

Conflicts of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Author contributions

Conceptualization, Sonia goyal and Amrit Kaur; methodology, Al_hussein M. Alturfi; software, Al_hussein M. Alturfi; validation, Al_hussein M. Alturfi; writing—original draft preparation, Al_hussein M. Alturfi; writing—review and editing, Al_hussein M. Alturfi; visualization, Al_hussein M. Alturfi; supervision, Sonia goyal and Amrit Kaur.

References

[1] D. Goldberg and J. Holland, "Genetic algorithms and machine learning", *Machine Learning* , Vol. 3, pp. 95–99, 1988.

[2] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces", *Journal*

of Global Optimization, Vol. 11, No. 4, pp. 341–359, 1997.

[3] R. Salgotra and U. Singh, "The naked mole-rat algorithm", *Neural Computing and Applications*, Vol. 31, No. 12, pp. 8837–8857, 2019.

[4] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer", *Advances in Engineering Software*, Vol. 69, pp. 46–61, 2014.

[5] L. Abualigah and A. Diabat, "Advances in sine cosine algorithm: a comprehensive survey", *Artificial Intelligence Review*, Vol. 54, No. 4, pp. 1–42, 2021.

[6] A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications", *Future Generation Computer Systems*, Vol. 97, pp. 849–872, 2019.

[7] L. Abualigah, D. Yousri, M. Elaziz, A. Ewees, M. A. Qaness, and A. Gandomi, "Aquila optimizer: A novel meta-heuristic optimization algorithm", *Computers & Industrial Engineering*, Vol. 157, p. 107250, 2021.

[8] P. Kusuma and A. Prasasti, "Guided Pelican Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 6, 2022, doi: 10.22266/ijies2022.1231.18.

[9] P. Kusuma and M. Kallista, "Stochastic Komodo Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No .4, 2022, doi: 10.22266/ijies2022.0831.15.

[10] P. D. Kusuma and A. Dinimaharawati, "Fixed Step Average and Subtraction Based Optimizer", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, pp. 339-351, 2022.

[11] Kusuma and F. Hasibuan, "Attack-Leave Optimizer: A New Metaheuristic that Focuses on The Guided Search and Performs Random Search as Alternative", *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 3, 2023, doi: 10.22266/ijies2023.0630.19.

[12] P. Kusuma and M. Kallista, "Quad Tournament Optimizer: A Novel Metaheuristic Based on Tournament Among Four Strategies", *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 2, 2023, doi: 10.22266/ijies2023.0430.22.

[13] P. Kusuma and A. Novianty, "Multiple Interaction Optimizer: A Novel Metaheuristic and Its Application to Solve Order Allocation Problem", *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 2, 2023, doi: 10.22266/ijies2023.0430.35.

[14] R. Salgotra and U. Singh, "Application of mutation operators to flower pollination algorithm", *Expert Systems with Applications*, Vol. 79, pp. 112–129, 2017.

[15] D. Karaboga, "An idea based on honey bee swarm for numerical optimization", *Tech. Rep., Technical Report-tr06, Erciyes University, Engineering Faculty, Computer*, Vol. 200, pp. 1-10, 2005.

[16] T. Ye, W. Wang, H. Wang, Z. Cui, Y. Wang, J. Zhao, and M. Hu, "Artificial bee colony algorithm with efficient search strategy based on random neighborhood structure", *Knowledge-Based Systems*, Vol. 241, p. 108306, 2022.

[17] D. Ustun, A. Toktas, U. Erkan, and A. Akdagli, "Modified artificial bee colony algorithm with differential evolution to enhance precision and convergence performance", *Expert Systems with Applications*, Vol. 198, p. 116930, 2022.

[18] E. Kaya, B. Gorkemli, B. Akay, and D. Karaboga, "A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems", *Engineering Applications of Artificial Intelligence*, Vol. 115, p. 105311, 2022.

[19] B. Akay, D. Karaboga, B. Gorkemli, and E. Kaya, "A survey on the artificial bee colony algorithm variants for binary, integer and mixed integer programming problems", *Applied Soft Computing*, Vol. 106, p. 107351, 2021.

[20] A. Sharma, A. Sharma, S. Choudhary, R. K. Pachauri, A. Shrivastava, and D. Kumar, "A review on artificial bee colony and it's engineering applications", *Journal of Critical Reviews*, Vol. 7, No. 11, pp. 4097–4107, 2020.

[21] E. Kaya, B. Gorkemli, B. Akay, and D. Karaboga, "A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems", *Engineering Applications of Artificial Intelligence*, Vol. 115, p. 105311, 2022.

[22] T. Ye, H. Wang, W. Wang, T. Zeng, L. Zhang, and Z. Huang, "Artificial bee colony algorithm with an adaptive search manner and dimension perturbation", *Neural Computing and Applications*, Vol. 34, No. 19, pp. 16239-16253, 2022.

[23] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems", *International Journal of Industrial Engineering Computations*, Vol. 7, No. 1, pp. 19–34, 2016.

[24] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization", *KanGAL Report*, p. 2005005, 2005.

[25] H. Wang, C. Liu, H. Wu, B. Li, and X. Xie, “Optimal pattern synthesis of linear array and broadband design of whip antenna using grasshopper optimization algorithm”, *International Journal of Antennas and Propagation*, Vol. 2020, p. 14, 2020.

[26] A. Das, D. Mandal, S. Ghoshal, and R. Kar, “Moth flame optimization based design of linear and circular antenna array for side lobe reduction”, *International Journal of Numerical Modelling: Electronic Networks Devices and Fields*, Vol. 32, No. 1, p. e2486, 2019.

[27] E. Owoola, K. Xia, T. Wang, A. Umar, and R. Akindele, "Pattern synthesis of uniform and sparse linear antenna array using mayfly algorithm”, *IEEE Access*, Vol. 9, pp. 77954-77975, 2021.

[28] N. Dib, “Design of linear antenna arrays with low side lobes level using symbiotic organisms search”, *Progress In Electromagnetics Research B*, Vol. 68, pp. 55–71, 2016.

[29] K. Erhan, S. Basbug, and K. Guney, "Linear antenna array synthesis by modified seagull optimization algorithm”, *The Applied Computational Electromagnetics Society Journal (ACES)*, Vol. 36, pp. 1552-1562, 2021.

[30] M. Khodier and M. A. Aqeel, “Linear and circular array optimization: A study using particle swarm intelligence”, *Progress In Electromagnetics Research B*, Vol. 15, pp. 347–373, 2009.

[31] A. Durmus, “Novel metaheuristic optimization algorithms for sidelobe suppression of linear antenna array”, In: *Proc. of 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 291-294, 2021.

[32] M. Khodier, “Comprehensive study of linear antenna array optimization using the cuckoo search algorithm”, *IET Microwaves, Antennas & Propagation*, Vol. 13, No. 9, pp. 1325–1333, 2019.

[33] U. Singh and R. Salgotra, “Synthesis of linear antenna arrays using enhanced firefly algorithm”, *Arabian Journal for Science and Engineering*, Vol. 44, pp. 1961–1976, 2019.

[34] R. Salgotra and U. Singh, “A novel bat flower pollination algorithm for synthesis of linear antenna arrays”, *Neural Computing and Applications*, Vol. 30, pp. 2269–2282, 2018.

[35] G. Sun, Y. Liu, H. Li, S. Liang, A. Wang, and B. Li, “An antenna array sidelobe level reduction approach through invasive weed optimization”, *International Journal of Antennas and Propagation*, Vol. 2018, pp. 1–16, 2018.

[36] P. Lakshman and D. Ghosh, “Linear antenna array synthesis using cat swarm optimization”, *AEU International Journal of Electronics and Communications*, Vol. 68, No. 6, pp. 540-549, 2014.

[37] A. Sharaqa, “Biogeography-based optimization method and its application in electromagnetics”, *Master Thesis*, Jorden University for Science and Technology, 2012.

[38] J. Derrac, S. Garc´ıa, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms”, *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp. 3–18, 2011.

[39] R. Salgotra, U. Singh, S. Singh, and N. Mittal, “A hybridized multi-algorithm strategy for engineering optimization problems”, *Knowledge-Based Systems*, Vol. 217, p. 106790, 2021.

[40] A. A. Azza, A. A. Jodah, and F. Harackiewicz, “Spider monkey optimization: A novel technique for antenna optimization”, *IEEE Antennas and Wireless Propagation Letters*, Vol. 15, pp. 1016–1019, 2015.