A comparison of swarm-based optimization algorithms in linear antenna array synthesis

Ali Durmus1 · Rifat Kurban2 · Ercan Karakose3

Received: 12 January 2021 / Accepted: 17 April 2021 / Published online: 4 May 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Today, the design of antenna arrays is very important in providing effective and efficient wireless communication. The purpose of antenna array synthesis is to obtain a radiation pattern with a low side lobe level (SLL) at a desired half power beam width in far-field. The amplitude and position values of the array elements can be optimized to obtain a radiation pattern with suppressed SLLs. In this paper, swarm-based metaheuristic algorithms including particle swarm optimization (PSO), artificial bee colony (ABC), mayfly algorithm (MA) and jellyfish search (JS) are compared to determine the optimal design of linear antenna arrays. Extensive experiments are conducted on designing 10-, 16-, 24- and 32-element linear arrays by determining the amplitude and positions. Experiments are repeated 30 times due to the random nature of swarm-based optimizers, and statistical results show that the performance of the novel algorithms, MA and JS, is better than that of the well-known PSO and ABC methods.

Keywords Linear antenna array synthesis · Side lobe level · Particle swarm optimization · Artificial bee colony algorithm · Mayfly algorithm · Jellyfish search algorithm

1 Introduction

With the introduction of Industry 4.0 in daily life, the Internet of Things (IoT) has gained importance, and with this the need for wireless communication has increased in every field. In order to meet this need effectively and efficiently, advanced antenna designs are used. It is very difficult to obtain the desired radiation pattern with a single antenna. Therefore, antenna arrays are formed by bringing together more than one antenna. In this way, both routing and communication demand over long distances is effectively satisfied. Antennas with different geometries are designed to obtain the desired radiation pattern, and are designated as linear, circular, elliptical and rectangular antennas according to their geometric structures [1].

In the design of an effective and efficient antenna array, the aim is to suppress the side lobe levels (SLL) in the radiation pattern and to make the half power beam width (HPBW) as narrow as possible. By preventing electromagnetic interference in the environment, the side lobes in the radiation pattern are suppressed in order to receive only the data in the desired direction, and the HPBW is narrowed in order to communicate with less loss with long distances [2].

There are several studies in the literature on linear antenna array (LAA) synthesis, which is one of the most widely studied topics in electromagnetic problems. Generally, in antenna array synthesis, three parameters are adjusted by optimization algorithms to obtain the desired SLL and HPBW. These are the amplitudes, positions and phases of the array elements. Different methods are used to determine these parameter values in antenna arrays. Classical methods are generally derivative-based, and due to their computational difficulty, these methods have been replaced by metaheuristic algorithms, which are faster and more flexible. There are several metaheuristic optimization techniques for LAA synthesis in the literature. A plant growth simulation algorithm was applied by Guney et al. [3] to pattern nulling of LAAs by amplitude-only control, while Dib et al. [4] proposed a symbiotic organism search (SOS) technique.
for synthesis of LAAs. A biogeography-based optimization (BBO) was utilized for the design of linear and elliptical antenna arrays [5], and Khodier and Al-Aqeel [6] proposed a particle swarm optimization (PSO) algorithm for linear and circular array design. Taguchi’s optimization method and self-adaptive differential evolution were used by Dib et al. to design LAAs [7]. Das et al. implemented moth flame optimization (MFO) for the synthesis of linear and circular antenna arrays for side lobe reduction [8]. Subhashini [9] applied a runner-root algorithm to control the SLL and null depths in LAAs. The artificial bee colony (ABC) algorithm was also utilized to design LAAs [10, 11]. The cuckoo search (CS) algorithm was used to design LAA by minimizing the maximum SLL with and without null steering [12]. The pattern synthesis of LAA is realized using an improved differential evolution algorithm reported by Zhang et al. [13]. Three kinds of antenna arrays, i.e. LAA, circular antenna array (CAA) and random antenna array (RAA), were studied in [14] with chicken swarm optimization (CSO). The strawberry algorithm (SBA) has been applied as an optimization tool for antenna array synthesis, and faster convergence characteristics are obtained in addition to improving side lobe suppression and prescribed null placements [15].

The no-free-lunch (NFL) theorem in optimization problems states that, for certain applications, the computational cost of finding an averaged solution for all problems in that class is the same for any solution method. Therefore, no optimization algorithm offers the best solution [16], and new algorithms are thus continuously developed and applied to optimization problems such as antenna array design.

Based on a literature review, although many different metaheuristic optimization methods are used in LAA synthesis, no study is found in which swarm-based algorithms—which constitute an important part of metaheuristic algorithms—are statistically evaluated together. Investigation of this subject is very important in terms of objectively evaluating the performance and speed of known algorithms and algorithms newly introduced to the literature.

In this study, LAAs with different numbers of elements are considered. The objective of this study is to statistically compare the performance of swarm-based optimization algorithms in linear antenna array design. Two of these swarm-based optimization algorithms chosen for this purpose are PSO and ABC, which are well known in the literature and have many applications in this field. The other two algorithms are the mayfly algorithm (MA) and jellyfish search (JS) algorithm, which to the best of our knowledge are implemented in our work for the first time in this field. The common feature of these four algorithms is that they are all inspired by the social behavior of different species of animals, fish, insects and bee swarms. To test the performance of these metaheuristic methods in LAA synthesis, 10-, 16-, 24- and 32-element arrays are examined. Using these algorithms, the optimal amplitude and position values of the LAA with different numbers of elements are found. In addition, the values obtained by the algorithms for SLL and CPU timing are statistically compared.

The rest of this paper is organized as follows: The problem formulation is examined in Sect. 2. In Sect. 3, the swarm-based metaheuristic optimization methods are briefly described. Numerical results and comparative statistical data obtained with PSO, ABC, MA and JS are given in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Linear array model and problem formulation

The configuration of the linear array with 2M elements placed symmetrically along the x-axis is shown in Fig. 1 [17]. With reference to the origin point, a total of 2M elements of the array are placed equally in both regions of the x-axis as M elements. The array factor (AF) expression of the LAA with symmetrical M elements positioned along the x-axis is given as follows.

$$\text{AF}(\theta) = 2 \sum_{n=1}^{M} I_n \cos(kd_n \sin \theta + \alpha_n)$$

(1)

where $\alpha_n$, $I_n$ and $d_n$ are phase, excitation amplitude and position weight of nth element in array, respectively. The scanning angle is $\theta$, $k$ represents the wave number and is formulated as $k = 2\pi / \lambda$. The total number of elements in the antenna array is 2M, but since these antennas are symmetrical, the parameter to be optimized is equal to half of the total number of antennas, $M$. In this study, $\alpha_n$ phase values are taken as zero.

![Fig. 1 Geometry of M-element LAA [17]](image-url)
The position and excitation amplitude values of the LAAs with the desired radiation patterns are optimally determined by four swarm-based algorithms, namely PSO, ABC, MA and JA. The purpose of antenna design is to transfer data over long distances without being affected by the electromagnetic pollution in the environment. Therefore, it is very important to suppress unwanted interference in the environment. In order to prevent these attempts, the SLLs of the radiation patterns should be suppressed. The aim here is to obtain a radiation model with a minimum SLL value. The HPBW value is also included in the fitness function, and this value has been determined as constant in all simulation studies. To achieve the desired goal, the fitness function can be defined as follows.

\[
\text{Fitness}_{\text{function}} = \begin{cases} 
\text{Inf} & \text{if } \text{HPBW}_{\text{current}} > \text{HPBW}_{\text{desired}} \\
\text{f}_{\text{SLL}} & \text{else}
\end{cases}
\]  

where \( \text{HPBW}_{\text{current}} \) is the HPBW of the potential solution produced by the optimization algorithms, \( \text{HPBW}_{\text{desired}} \) is the desired HPBW for the specified array. \( \text{Inf} \) is a penalty value and \( \text{f}_{\text{SLL}} \) is the function formulated as
\[
\text{f}_{\text{SLL}} = \max\{\text{AF}(\theta)\}
\]

### 3 Swarm-based optimization techniques

#### 3.1 Particle swarm optimization (PSO)

PSO is a swarm intelligence algorithm developed in 1995 based on the population-based movement of birds or fish and is used to solve optimization problems. In PSO, a search field is defined and the position of a moving particle in that field represents the solution to the problem. In problem-solving, each particle changes its position according to its choice and the choice of the community with which it moves together. The position of any particle is changed by adding a speed to its position. Thus, the velocity of each particle depends on its previous most optimal position and the previous most optimal position of the flock to which it is attached. The advantages of PSO can be briefly summarized as follows: the derivative is not calculated so that the solution is not stuck at a local optimum where the derivative is zero; the members in the group share information constantly between one another; and the knowledge of the best solutions is used by all members. More information about the PSO algorithm can be found in Kennedy and Eberhart [18]. Figure 2 shows the application of the PSO algorithm for the synthesis of optimal LAAs.

![Implementation flowchart of PSO to LAA design](image)

**Fig. 2** Implementation flowchart of PSO to LAA design

#### 3.2 Artificial bee colony (ABC) algorithm

The ABC algorithm is a flock of intelligence methods that have been successfully applied to many optimization problems and proposed in 2007. In the ABC algorithm, bees are divided into three classes: employed, onlooker, and scout. The ABC algorithm is inspired by the foraging behavior of bees. Employed bees take advantage of specific food sources and are on the move to find better food sources. While the onlooker bees watch the movements of the employed bees to select their food sources, the scout bees seek out random food sources. In this algorithm, bees using the food source represent an existing solution and lead to the resolution of new problems by simulating how the food source will be consumed in the audience bee community. The working logic of the ABC algorithm has four stages. At each renewal, these steps are performed sequentially by running, and the iteration repeats until they reach the goal. Details on the ABC method can be found in Karaboga and Basturk [19]. Implementation of the ABC algorithm for LAA design problems is given in Fig. 3.
3.3 Mayfly algorithm (MA)

The MA is a novel metaheuristic method which was proposed in 2020, inspired by the mating behavior of the mayfly to ensure optimal breeding success. A mayfly’s lifespan is only a few days, and according to this algorithm, mayflies become adult after hatching, and the strongest mayflies are considered to survive. A potential solution for any problem corresponds to the position of any mayfly in the search space. In the algorithm, two different clusters corresponding to male and female populations are formed. That is, each male or female mayfly is randomly placed into the problem space in an area represented by a $d$-dimensional vector $x = (x_1, \ldots, x_d)$. The $x$ vector performance is evaluated on the predetermined target function $f(x)$. Simply stated, the velocity of a mayfly can be defined as the change in position $v = (v_1, \ldots, v_d)$. The flight orientation of a mayfly is expressed as a complex that includes both the collective and individual flight preferences of the flies. That is, each mayfly orientation is taken and used as the individual best position ($best$) and the good position ($gbest$) in swarm movement. Readers can find detailed information about the MA in Zervoudakis and Tsafarakis [20]. A flowchart of the MA algorithm to solve LAA design problems is given in Fig. 4.

4 Jellyfish search (JS) algorithm

The JS algorithm, inspired by the behavior of jellyfish in the ocean, is one of the newest swarm intelligence metaheuristic methods for optimization problems, developed in 2020. The behavior simulation of the jellyfish in the swarm includes the movement of the jellyfish in the flock, the instant tracking of the sea current and the time control mechanism for
movements. JS provides an optimum convergence by balancing the exploration and utilization of the search area. The jellyfish’s movement in the sea is based on searching for food. For this reason, jellyfish constantly move towards areas with more food. Detailed information on the JS optimization method can be found in Chou and Truong [21]. Figure 5 shows a flowchart of the JS algorithm to design an optimal solution for LAAs.

5 Numerical results

In this study, the amplitude and position of LAAs with 10, 16, 24 and 32 elements are optimally determined by four different swarm-based optimization methods, and the values found are compared statistically. The common feature of the selected PSO, ABC, MA and JS algorithms is that they are all swarm-based. The reason why PSO and ABC are preferred in simulation studies is that these algorithms are the most widely used optimization techniques in the literature. MA and JS are optimization techniques recently presented to the literature, and in this study these algorithms have been applied to antenna array synthesis for the first time.

The purpose of all experiments is to reach radiation patterns with lower SLL values at a desired HPBW. Experimental simulations are realized using MATLAB software and a personal computer with 16 GB of RAM and a 2.6 GHz i7 processor.

For all algorithms, the population size is set at 50 and the maximum iteration number is set at 500. The remaining internal parameters of the optimization algorithms used in the experiments are given in Table 1. These parameters are the suggested default values from the creators of the original algorithms.

---

Fig. 5 Implementation flowchart of JS to LAA design
Since the following HPBW values are commonly chosen in the majority of the studies in literature, the desired HPBW are determined as 12.5°, 9°, 6° and 3.5° for 10-, 16-, 24- and 32-element arrays, respectively.

Due to a lack of space, several unreported experiments show that after 500 iterations, there is no significant improvement in the fitness function for all optimization algorithms used in this paper.

On the other hand, studies in the literature are examined in the statistical evaluation of non-deterministic metaheuristic optimization algorithms. Due to their random nature, it is seen that the same experiment is run independently between 20 and 100 times with different random number seeds under the same conditions in several studies. Since this value is determined as 30 from the original papers of the algorithms that we used in our study, it is considered that running the experiments 30 times would be sufficient. Average, best and worst conditions and the standard deviation of the SLL and CPU computation times of all runs are described extensively in the following experiments.

5.1 Amplitude-only design of optimal LAA

In the first example, the amplitudes of the 10-, 16-, 24- and 32-element symmetric LAA arrays are determined using the PSO, ABC, MA and JS methods to achieve the optimal design. Figure 6 shows the radiation patterns obtained with the PSO, ABC, MA and JS methods to achieve the optimal design. As can be clearly seen from Table 2 and Fig. 6, for the 10-element LAA, all algorithms except ABC performed the same in terms of mean SLL values. JS is ranked first in terms of CPU time consumption, solving the problem in 3.1 s.

### Table 1 Parameter settings of the optimization algorithms

| Algorithm | Parameters |
|-----------|------------|
| PSO       | $w = 0.5$, $w_{damp} = 1$, $c_1 = 1$, $c_2 = 1$ |
| ABC       | onlooker_count = 25, max_acc = 1 |
| MA        | $g = 0.5$, $g_{damp} = 1$, $a_1 = 1.5$, $a_2 = 1.5$, beta = 2, dance = 1, $f_1 = 1$, dance_damp = 0.8, $f_1$ = 0.99, mu = 0.01 |
| JS        | No parameters |

![Fig. 6](image-url) Best radiation patterns for 10, 16, 24 and 32 elements for amplitude-only optimal LAA design
PSO, MA and ABC are ranked second, third and fourth, solving the problem in 9.7, 13.3 and 14.2 s, respectively. For the 16-element LAA, the algorithms MA, JS, PSO and ABC are ranked from best to worst, respectively, in terms of mean SLL, and MA, JS, ABC and PSO in terms of the standard deviation of SLL. JS, PSO, ABC and MA solved the problem in 4.4, 10.2, 15.6 and 17.7 s, respectively. For the 24-element LAA, MA, JS, PSO and ABC are ranked best to worst, respectively, in terms of mean SLL, and MA, ABC, JS and PSO in terms of the standard deviation of SLL. JS, PSO, ABC and MA solved the problem in 6.1, 11.9, 18.4 and 23.7 s, respectively. For the 32-element LAA, JS, MA, PSO and ABC are ranked from best to worst in terms of mean SLL, and JS, MA, ABC and PSO in terms of the standard deviation of SLL. JS, PSO, ABC and MA solved the problem in 8.1, 13.7, 21.1 and 29.3 s, respectively. As a result, for amplitude-only design of optimal 10-, 16, 24 and 32 elements are tabulated in Table 3.

The averaged convergence curves of PSO, ABC, MA and JS methods for 10, 16, 24 and 32 elements are illustrated in Fig. 7 for 10-, 16-, 24- and 32-element amplitude-only optimization of LAA. As can be seen from Fig. 7, for the 10-element LAA, all algorithms except ABC converged to the best result in 200 iterations. For the 16-element LAA, PSO and MA converged early, but JS caught up after 350 iterations. For the 24-element LAA, MA converged smoothly compared to the other algorithms. For the 32-element LAA, MA converged early, but JS took the lead at the end.

### 5.2 Position-only design of optimal LAA

In the second example, the positions of the 10-, 16-, 24- and 32-element symmetric LAA are determined using the PSO, ABC, MA and JS methods for 10, 16, 24 and 32 elements are tabulated in Table 3.

The average convergence curves of PSO, ABC, MA and JS methods for 10, 16, 24 and 32 elements are illustrated in Fig. 7 for 10-, 16-, 24- and 32-element position-only optimization of LAA. As can be seen from Fig. 7, for the 10-element LAA, all algorithms except ABC converged to the best result in 200 iterations. For the 16-element LAA, PSO and MA converged early, but JS caught up after 350 iterations. For the 24-element LAA, MA converged smoothly compared to the other algorithms. For the 32-element LAA, MA converged early, but JS took the lead at the end.

### 5.2 Position-only design of optimal LAA

In the second example, the positions of the 10-, 16-, 24- and 32-element symmetric LAA are determined using the PSO, ABC, MA and JS methods to achieve the optimum design. Figure 8 shows the radiation patterns obtained with PSO, ABC, MA and JS. The suppressed SLL and CPU times obtained by using PSO, ABC, MA and JS are listed in Table 4. As can be clearly seen from Table 4 and Fig. 8, for element sizes, the MA, JS, PSO and ABC algorithms are ranked best to worst in terms of mean SLL values for 30 runs. With regard to CPU times, JS, PSO, ABC and MA are ranked from best to worst. For position-only design of the optimal LAA, MA performed the best among

### Table 2 Statistical SLL and CPU time comparison of PSO, ABC, MA and JS for amplitude-only design of optimal LAA for 10, 16, 24 and 32 elements

| Element | PSO | ABC | MA | JS |
|---------|-----|-----|----|----|
| 10-element |     |     |    |    |
| Best SLL (dB) | −26.9772 | −26.9772 | −26.9772 | −26.9772 |
| Mean SLL (dB) | −26.9772 | −26.9772 | −26.9772 | −26.9772 |
| Worst SLL (dB) | −26.9772 | −26.9772 | −26.9772 | −26.9772 |
| Std. dev. SLL (dB) | 0.0001 | 0.0001 | 0.0001 | 0.0001 |
| Best CPU time (s) | 7.7581 | 13.8050 | 12.6659 | 3.1369 |
| Mean CPU time (s) | 9.7063 | 14.2248 | 13.3780 | 3.3126 |
| Worst CPU time (s) | 10.8955 | 14.8900 | 14.0229 | 3.6738 |
| Std. dev. CPU time (s) | 0.5090 | 0.2599 | 0.3097 | 0.1286 |

| 16-element |     |     |    |    |
| Best SLL (dB) | −40.2334 | −39.7988 | −40.2472 | −40.2471 |
| Mean SLL (dB) | −39.7041 | −38.8234 | −40.2472 | −40.1986 |
| Worst SLL (dB) | −35.9071 | −37.8194 | −40.2472 | −39.1132 |
| Std. dev. SLL (dB) | 0.8996 | 0.4736 | 0.0000 | 0.2065 |
| Best CPU time (s) | 9.5540 | 14.7093 | 16.7023 | 4.1206 |
| Mean CPU time (s) | 10.1806 | 15.6192 | 17.3171 | 4.3870 |
| Worst CPU time (s) | 10.9683 | 16.4124 | 18.6868 | 4.7096 |
| Std. dev. CPU time (s) | 0.4174 | 0.5498 | 0.6040 | 0.1951 |

| 24-element |     |     |    |    |
| Best SLL (dB) | −40.5498 | −39.7922 | −41.2720 | −40.9730 |
| Mean SLL (dB) | −38.6388 | −38.6031 | −41.0523 | −38.6710 |
| Worst SLL (dB) | −33.0826 | −37.3883 | −40.4024 | −32.1808 |
| Std. dev. SLL (dB) | 1.7683 | 0.6543 | 0.2913 | 1.7260 |
| Best CPU time (s) | 11.5868 | 17.9760 | 23.0663 | 5.9064 |
| Mean CPU time (s) | 11.9383 | 18.3983 | 23.7065 | 6.0734 |
| Worst CPU time (s) | 14.0326 | 20.0984 | 26.6039 | 6.6417 |
| Std. dev. CPU time (s) | 0.4680 | 0.4938 | 0.9999 | 0.1594 |

| 32-element |     |     |    |    |
| Best SLL (dB) | −22.7572 | −22.4181 | −23.2519 | −23.1101 |
| Mean SLL (dB) | −21.9665 | −21.9189 | −22.8378 | −22.8525 |
| Worst SLL (dB) | −21.0074 | −21.5110 | −22.2075 | −22.3647 |
| Std. dev. SLL (dB) | 0.4946 | 0.2199 | 0.2187 | 0.1526 |
| Best CPU time (s) | 12.8456 | 19.9022 | 27.2495 | 7.5898 |
| Mean CPU time (s) | 13.6918 | 21.0755 | 29.3229 | 8.1332 |
| Worst CPU time (s) | 15.0637 | 23.0044 | 31.6852 | 8.8115 |
| Std. dev. CPU time (s) | 0.4680 | 0.4938 | 0.9999 | 0.1594 |
Table 3  The best optimized amplitude values of LAA using PSO, ABC, MA and JS methods for 10, 16, 24 and 32 elements

| Element Count | PSO Values                                      | ABC Values                                      | MA Values                                      | JS Values                                      |
|---------------|------------------------------------------------|------------------------------------------------|------------------------------------------------|------------------------------------------------|
| 10-Element    | [0.8900 0.7928 0.6231 0.4220 0.2950]            | [0.8859 0.7895 0.6204 0.4204 0.2939]            | [0.9837 0.8763 0.6888 0.4664 0.3261]            | [0.8886 0.7916 0.6222 0.4213 0.2945]            |
| 16-Element    | [0.9992 0.9342 0.8147 0.6591 0.4900 0.3294 0.1941 0.1114] | [1.0000 0.9316 0.8166 0.6597 0.4891 0.3322 0.1953 0.1108] | [0.8406 0.7859 0.6853 0.5544 0.4122 0.2770 0.1632 0.0936] | [0.9515 0.8896 0.7757 0.6276 0.4665 0.3135 0.1847 0.1060] |
| 24-Element    | [0.8934 0.8600 0.8219 0.7357 0.6627 0.5556 0.4607 0.3594 0.2726 0.1818 0.1362 0.0889] | [1.0000 0.9776 0.9121 0.8359 0.7436 0.6368 0.5135 0.4047 0.3097 0.2208 0.1396 0.1095] | [0.9003 0.8741 0.8224 0.7503 0.6625 0.5640 0.4626 0.3633 0.2708 0.1906 0.1238 0.0993] | [0.8931 0.8667 0.8163 0.7426 0.6579 0.5610 0.4581 0.3599 0.2697 0.1905 0.1230 0.0973] |
| 32-Element    | [0.6700 0.7960 0.6850 0.6937 0.6008 0.7223 0.5845 0.5345 0.5220 0.5177 0.4727 0.3034 0.4385 0.2449 0.4295 0.5994] | [1.0000 1.0000 0.9863 1.0000 0.8747 0.9413 0.7950 0.8109 0.7527 0.6417 0.6608 0.5093 0.5174 0.4444 0.5696 0.8410] | [0.7611 0.7519 0.7299 0.7333 0.6923 0.6630 0.6431 0.5801 0.5626 0.5083 0.4555 0.4302 0.3451 0.3391 0.2942 0.7365] | [0.5929 0.5858 0.5860 0.5705 0.5142 0.5672 0.4624 0.4725 0.4499 0.3727 0.3748 0.3433 0.2509 0.2700 0.2631 0.5525] |

Fig. 7  Averaged convergence curves of PSO, ABC, MA and JS for amplitude-only design of LAA with 10, 16, 24 and 32 elements
the four algorithms in terms of mean SLL. However, MA is the slowest algorithm among the four in terms of CPU time consumption. JS performed second in terms of mean SLL; however, JS is the fastest of the four algorithms in all simulations. The best position values obtained by the PSO, ABC, MA and JS methods for 10, 16, 24 and 32 elements are tabulated in Table 5.

Averaged convergence curves of PSO, ABC, MA and JS of 30 independent runs are illustrated in Fig. 9 for 10, 16, 24 and 32 element position-only optimization of LAA. As can be seen from Fig. 9, for all LAA element sizes, MA and JS converged better than PSO and ABC.

5.3 Implementation

In Fig. 10, a possible practical implementation of LAAs is shown. This figure may help the reader whose goal is to realize such arrays using swarm-based optimization algorithms. According to the desired HPBW of the radiation pattern, the optimization method tunes the position controllers and the amplitude weights to arrange the inter-element spacing and the current excitation of the array elements, respectively. Therefore, it is necessary to suppress the side lobe level (SLL) in the radiation pattern to prevent interference from systems operating in the same frequency band. According to the rapidly changing demands of the application, it is possible to direct the main beam to the desired direction faster and more effectively with swarm-based algorithms.

6 Conclusions

In this paper, the optimal design of LAAs with different numbers of elements is realized using the PSO, ABC, MA and JS swarm-based optimization methods. The position and amplitude values of the antenna array elements are determined for the first time by novel MA and JS techniques to obtain radiation patterns with the lowest SLL at a desired HPBW. Experiments are conducted for 10-, 16-, 24- and 32-element LAA designs for both amplitude-only and position-only cases. Simulations are repeated 30 times due to the random nature of the swarm-based metaheuristic
algorithms. The mean of the maximum SLL values and the standard deviations of independent runs are compared, along with the CPU consumption times of the algorithms. Results show that MA performs the best and JS the second best in terms of mean SLL in general for all cases. However, MA is the slowest and JS is the fastest algorithm with regard to time consumption. The overall average CPU time for MA is 20.9 s and for JS is 5.5 s. Therefore, we can say that MA consumes ~3.8 times more energy than the JS algorithm. In conclusion, the results for the novel MA and JS algorithms are better than those for the well-known PSO and ABC optimization techniques. In future studies, the MA and JS algorithms can be used for the synthesis of antenna arrays with elliptical, circular and concentric circular geometries.

### Table 4 Statistical SLL and CPU time comparison of PSO, ABC, MA and JS for position-only design of optimal LAA for 10, 16, 24 and 32 elements

|            | 10-element |            | 16-element |
|------------|------------|------------|------------|
|            | PSO        | ABC        | MA         | JS         | PSO        | ABC        | MA         | JS         |
| Best SLL (dB) | −21.3920  | −21.1796   | −21.3958   | −21.4305   | −24.5285   | −22.7773   | −24.3570   | −23.5265   |
| Mean SLL (dB) | −20.2687  | −19.7332   | −20.7130   | −20.6417   | −22.5863   | −22.3006   | −22.9624   | −22.7405   |
| Worst SLL (dB) | −19.7709  | −17.6890   | −19.0663   | −19.0669   | −19.9370   | −21.3002   | −20.1059   | −21.2464   |
| Std. Dev. SLL (dB) | 1.0769    | 0.3094     | 0.9393     | 1.1319     | 1.0299     | 0.3189     | 0.8128     | 0.6167     |
| Best CPU time (s) | 8.7570    | 12.8468    | 13.8618    | 2.9977     | 9.6604     | 14.6433    | 16.2801    | 4.2945     |
| Mean CPU time (s) | 9.0856    | 13.6510    | 14.4751    | 3.1230     | 10.2636    | 15.5047    | 17.3993    | 4.5102     |
| Worst CPU time (s) | 9.4691    | 14.6641    | 15.2326    | 3.3974     | 11.0657    | 16.3744    | 18.3763    | 4.7998     |
| Std. Dev. CPU time (s) | 0.2058    | 0.3509     | 0.4181     | 0.0936     | 0.3686     | 0.5543     | 0.6421     | 0.1525     |

### Table 5 The best optimized position values of LAA using the PSO, ABC, MA and JS methods for 10, 16, 24 and 32 elements

|            | 10-element |            | 16-element |
|------------|------------|------------|------------|
|            | PSO        | ABC        | MA         | JS         | PSO        | ABC        | MA         | JS         |
| 10-element | [0.3345 0.3888 0.9681 1.3514 2.0090] | [0.3683 0.3683 0.9831 1.3721 2.0396] | [0.3600 0.3600 0.9646 1.3439 1.9999] | [0.3658 0.3783 0.9967 1.3876 2.0654] | [0.1823 0.4752 0.6727 1.1865 1.3424 1.8701 2.3206 3.0066] | [0.1851 0.5481 0.9513 1.3352 1.8037 2.2825 2.9197 3.6633] | [0.1112 0.5421 0.6335 1.1725 1.3534 1.8597 2.3176 2.9917] | [0.2002 0.4846 0.8059 1.1563 1.5154 1.9532 2.4831 3.1557] |
| 16-element | [0.0597 0.5555 0.6413 1.1163 1.3584 1.7007 2.1020 2.4342 2.9100 3.3799 4.0396 4.7953] | [0.1733 0.6391 0.9411 1.4267 1.7968 2.2816 2.7032 3.2638 3.7707 4.4611 5.2854 6.0878] | [0.1309 0.4822 0.6646 1.1426 1.2523 1.7255 2.0052 2.4121 2.8350 3.3138 3.9515 4.7043] | [0.1802 0.4880 0.8975 1.2023 1.6218 1.9480 2.4049 2.8536 3.3207 3.9171 4.6549 5.4438] |
| 24-element | [0.2119 0.6917 0.9307 1.4255 1.8137 2.2378 2.7201 3.1631 3.6071 4.1347 4.7124 5.2376 5.9530 6.7527 7.6710 8.4690] | [0.2120 0.5839 0.9696 1.2748 1.7577 2.0859 2.5501 3.0111 3.4238 3.9094 4.4924 4.9714 5.6404 6.4386 7.2759 8.0211] | [0.2029 0.5257 0.9317 1.3053 1.7267 2.1467 2.5052 2.9446 3.4058 3.9146 4.4395 4.9265 5.5611 6.4199 7.2074 8.1295] | [0.1835 0.5751 0.9457 1.3239 1.7507 2.1134 2.5725 2.9558 3.4466 3.8855 4.4722 4.9815 5.5849 6.4261 7.2671 8.1108] |
example, using monopole, dipole, patch, slot, spiral, horn, helix or loop elements. Since antenna arrays are applied in many different fields, including the defense industry, satellite communication, biomedical applications and wireless personal communication, novel antenna array design methods are still needed.

References

1. Balanis, C.A.: Antenna Theory: Analysis and Design. Wiley (2016)
2. Mailloux, R.J.: Phased Array Antenna Handbook. Artech house (2017)
3. Guney, K., Durmus, A., Basbug, S.: A plant growth simulation algorithm for pattern nulling of linear antenna arrays by amplitude control. Prog. Electromagn. Res. 17, 69–84 (2009)
4. Dib, N.I.: Design of linear antenna arrays with low side lobes level using symbiotic organisms search. Prog. Electromagn. Res. 68, 55–71 (2016)
5. Sharaq, A., Dib, N.: Design of linear and elliptical antenna arrays using biogeography based optimization. Arab. J. Sci. Eng. 39(4), 2929–2939 (2014)
6. Khodier, M.M., Al-Aqeel, M.: Linear and circular array optimization: a study using particle swarm intelligence. Prog. Electromagn. Res. 15, 347–373 (2009)
7. Dib, N.I., Goudos, S.K., Muhsen, H.: Application of Taguchi’s optimization method and self-adaptive differential evolution to
the synthesis of linear antenna arrays. Prog. Electromagn. Res. **102**, 159–180 (2010)
8. Das, A., Mandal, D., Ghoshal, S.P., Kar, R.: Moth flame optimization based design of linear and circular antenna array for side lobe reduction. Int. J. Numer. Model. Electron. Netw. Dev. Fields **32**(1), e2486 (2019)
9. Subhashini, K.R.: Runner-root algorithm to control sidelobe level and null depths in linear antenna arrays. Arab. J. Sci. Eng. **45**(3), 1513–1529 (2020)
10. Zaman, M.A., Gaffar, M., Alam, M.M., Mamun, S.A., Matin, M.A.: Synthesis of antenna arrays using artificial bee colony optimization algorithm. Int. J. Microwave Opt. Technol. **6**(4), 234–241 (2011)
11. Basu, B., Mahanti, G.K.: Fire fly and artificial bees colony algorithm for synthesis of scanned and broadside linear array antenna. Prog. Electromagn. Res. **32**, 169–190 (2011)
12. Khodier, M.: Comprehensive study of linear antenna array optimization using the cuckoo search algorithm. IET Microw. Antennas Propag. **13**(9), 1325–1333 (2019)
13. Zhang, R., Zhang, Y., Sun, J., Li, Q.: Pattern synthesis of linear antenna array using improved differential evolution algorithm with SPS framework. Sensors **20**(18), 5158 (2020)
14. Liang, S., Fang, Z., Sun, G., Liu, Y., Qu, G., Zhang, Y.: Sidelobe reductions of antenna arrays via an improved chicken swarm optimization approach. IEEE Access **8**, 37664–37683 (2020)
15. Subhashini, K.R.: Antenna array synthesis using a newly evolved optimization approach: strawberry algorithm. J. Electr. Eng. **70**(4), 317–322 (2019)
16. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
17. Durmus, A., Kurban, R.: Optimum design of linear and circular antenna arrays using equilibrium optimization algorithm. Int. J. Microwave Wirel. Technol. 1–12 (2021)
18. Kennedy, J., Eberhart, R. Particle swarm optimization. In: Proceedings of ICNN’95-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE. (1995)
19. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007)
20. Zervoudakis, K., Tsafrakis, S.: A mayfly optimization algorithm. Comput. Ind. Eng. **145**, 106559 (2020)
21. Chou, J.-S., Truong, D.-N.: A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. Appl. Math. Comput. **389**, 125535 (2021)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.