Progress In Electromagnetics Research C, Vol. 60, 137–146, 2015

Genetical Swarm Optimizer for Synthesis of Multibeam Linear Antenna Arrays

Hichem Chaker*

Abstract—The paper presents a hybrid evolutionary algorithm suitable for the optimization of large-domain electromagnetic problems. The hybrid technique, called Genetical Swarm Optimization (GSO), combines Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). GSO algorithm is modelled on the concepts of Darwin's theory based on natural selection and evolution, and on cultural and social rules derived from the swarm intelligence. The problem is formulated and solved by means of the proposed algorithm. The examples are simulated to demonstrate the effectiveness and design flexibility of GSO in the framework of synthesis of multi-beam linear antennas arrays.

1. INTRODUCTION

Several evolutionary algorithms have been developed for optimization of every kind of electromagnetic problems. The general goal of the optimization is to find a solution that represents the global maximum of a fitness function. Electromagnetic optimization problems generally involve a large number of parameters, that can be either continuous, discrete, or both, and often include constraints in allowable values. In addition, the solution domain of electromagnetic optimization problems often has non differentiable and discontinuous regions, and often utilizes approximations or models of the true electromagnetic phenomena to conserve computational resources [1]. Global search methods have two competing goals, exploration and exploitation: exploration is important to ensure that every part of the solution domain is searched enough to provide a reliable estimate of the global optimum; exploitation, instead, is also important to concentrate the search effort around the best solutions found so far by searching their neighbourhoods to reach better solutions. Often global search methods are used together with other local search algorithm in order to improve efficiency and accuracy of the searching process. The evolutionary computation algorithms (EA) are stochastic optimization methods, which emulate biologic processes or natural phenomena [2]. The capability to find a global optimum, without being trapped in local optima, and the possibility to well face nonlinear and discontinuous problems, with great numbers of variables, are some advantages of these techniques. Besides these methods do not need to compute any derivatives in order to optimize the objective function and this fact allows to manage more complex fitness function. Moreover, in contrast with traditional searching methods, EAs do not depend strongly on the starting point [3]. Often a bad choice of the initial values can slow down the convergence of the entire process, or even drive the convergence towards a wrong solution, e.g., towards a local instead a global maximum or minimum. However, these algorithms have strong stochastic basis, therefore they need a lot of iterations to get a significant result, in particular when the optimization problem has a big number of unknowns [4, 5]. Considering Genetic Algorithms and Particle Swarm Optimization algorithms, most of the times, PSO have faster convergence rate than GA early in the run, but they are often outperformed by GA for long simulation runs, or when the number of unknowns increases. This is due to the different types of search, adopted by the two algorithms. The new hybrid

Received 2 November 2015, Accepted 7 December 2015, Scheduled 16 December 2015
* Corresponding author: Hichem Chaker (mh_chaker2005@yahoo.fr).
The author is with the Department of Electronic, University of Sidi Bel Abbès, Algeria.

technique here proposed, called Genetical Swarm Optimization, and consists in a strong co-operation of GA and PSO, since it maintains the integration of the two techniques for the entire run of simulation. In each iteration, in fact, some of the individuals are substituted by new generated ones by means of GA, while the remaining part is the same of the previous generation but moved on the solution space by PSO [6].

In this paper, we are interest in presenting the genetic swarm optimization method that will be applied to the synthesis of multibeam arrays. A big flexibility between features of the antennas array: amplitude and phase of feeding, ripple area, and secondary lobe level... is introduced.

2. ANTENNA ARRAY PATTERN FORMULATION

An array can form multiple narrow beams towards different directions. For example, suppose it is desired to form from two to four beams towards the steering angles 2, 3, and 4. The design of a linear array antenna is based on finding both magnitudes and phases excitation that can generate the desired patterns.

We consider a linear array of $2N$ isotropic antenna elements, which are assumed, uncoupled, symmetrically and equally spaced with half wavelength. Its array pattern can be described as follows as discussed by Ali [7]:

$$F(\theta) = 2\sum_{k=1}^{N} a_k \cos\left(\frac{2\pi}{\lambda} d_k \sin(\theta) + \delta_k\right) \tag{1}$$

where

N: Element number;

λ: Wavelength;

δ_k: Phases of the elements ($-180^\circ \leq \theta \leq 180^\circ$);

a_k: Amplitudes of the elements;

d_k: Distance between position of ith element and the array center;

θ: Scanning angle.

In our applications, we use an elementary source of square shape for substrate with the permittivity equal to 3.5, thickness equal to 0.159 cm and operating at 5 GHz.

In order to generate a beam pattern fulfilling some constraints, side lobe level lower than a fixed threshold or reproducing a desired shape, an array configuration must be synthesized. First of all, it is necessary to define the objective function that measures the difference between desired and synthesized beam pattern. Let us define a function called fitness function as follows:

$$\text{Fitness} = \text{Max} - \int_0^{\pi} |F_d(\theta) - F(\theta)|\, d\theta \tag{2}$$

The fitness function defined in Eq. (2) represents the general form for antenna pattern synthesis.

3. GENETIC ALGORITHM

Genetic Algorithm (GA) is one of the most effective evolutionary algorithms developed so far [8, 9]; it simulates the natural evolution, in terms of survival of the fittest, adopting pseudo-biological operators such as selection, crossover, mutation,and many other additional operators introduced to get a faster convergence rate. In GA, the set of parameters that characterizes a specific problem is called an individual or a chromosome and it is composed of a list of genes. Each gene contains the parameter itself or a suitable encoding of it. Each individual therefore represents a point in the search space, and hence a possible solution to the problem. The fitness function is therefore evaluated for each individual of the population, resulting in a score assigned to the individual. Based on this fitness score, a new population is generated iteratively with each successive population referred to as a generation [10]. Starting from a population of randomly generated individuals, the three basic GA operators (selection, crossover, and mutation) are applied in order to manipulate the genetic composition of this population.

Selection is the process by which the most highly rated individuals in the current generation are chosen to be involved as parents in the creation of a new generation. The crossover operator produces two new individuals (i.e., candidate solutions) by recombining the information from two parents. Crossover operation occurs in two steps. In the first one, a given number of crossing sites, along with the parent individual, are selected uniformly at random. In the second step, two new individuals are formed by exchanging alternate pairs of selection between the selected sites. The random mutation of some gene values in an individual is the third GA operator. Genetic Algorithms are very efficient at exploring the entire search space, but are relatively poor in finding the precise local optimal solution in the region in which the algorithm converges. Many efforts on the enhancement of traditional GAs have been proposed [11, 12], by modifying the structure of the population or the role that an individual plays in it (distributed GA, cellular GA, and symbiotic GA) or by modifying the basic operations of traditional GA, or by adding new ones, such as elitism [13, 14]. The Genetic Algorithm developed for this application uses real encoded genes, since for high number of variables they show themselves faster than binary ones to converge towards the maximum value [15]. Several additional operators have been developed for GA in order to get a faster convergence rate.

4. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is one of the more recently developed evolutionary techniques, and it is based on a suitable model of social interaction between independent agents (particles) and it uses social knowledge in order to find the global maximum of a generic function [16]. While for the GA, as shown in the last section, the improvement in the population fitness is assured by pseudo-biological operators, such as selection, crossover and mutation, the main PSO operator is the velocity update that takes into account the best position explored during the iterations, resulting in a migration of the swarm towards the global optimum. PSO is similar to the other evolutionary algorithms in that the system is initialized with a population of random solutions. Each potential solution, call particles, flies in the D-dimensional problems space with a velocity which is dynamically adjusted according to the flying experiences of its own and its colleagues. The location of the ith particle is represented as $X_i = (x_{i1}, \ldots, x_{id}, \ldots, x_{iD})$. The best previous position (which giving the best fitness value) of the ith particle is recorded and represented as $P_i = (p_{i1}, \ldots, p_{id}, \ldots, p_{iD})$, which is also called p_{best}. The index of the best p_{best} among all the particles is represented by the symbol g. the location Pg is also called g_{best}. The velocity for the ith particle is represented as $V_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{iD})$. The particle swarm optimization consists of, at each time step, changing the velocity and location of each particle toward its p_{best} and g_{best} locations according to the Equations (3) and (4) respectively:

$$v_{id} = w * v_{id} + c_1 * \text{rand}( ) * (p_{id} - x_{id}) + c_2 * \text{rand}( ) * (p_{gd} - x_{id}) \quad (3)$$

$$x_{id} = x_{id} + v_{id} \quad (4)$$

where w is inertia weight; c_1 and c_2 are acceleration constants; rand( ) is a random function in the range [0 1]. For Equation (3), the first part represents the inertia of previous velocity; the second part is the "cognition" part, which represents the private thinking by itself; the third part is the "social" part, which represents the cooperation among the particle. V_i is clamped to a maximum velocity $V_{\max} = (v_{\max,1}, \ldots, v_{\max,d}, \ldots, v_{\max,D})$. $V_{\max}$ determines the resolution with which regions between the present and the target positions are searched [17]. The process for implementation PSO is as follows:

a). Set current iteration generation $G_c = 1$. Initialize a population which including m particles, for the ith particle, it has random location X_i in specified space and for the dth dimension of V_i, $V_{id} = \text{rand}_2( ) \times V_{\max,d}$, where $\text{rand}_2( )$ is a random value in the range of $[-1\ 1]$;

b). Evaluate the fitness for each particle;

c). Compare the evaluated fitness value of each particle with its p_{best}. If the current value is better than p_{best}, and then set the current location as the p_{best} location. Furthermore, if current value is better than g_{best}, then reset g_{best} to the current index in particle array;

d). change the velocity and location of the particle according to the Equations (3) and (4), respectively;

e). $G_c = G_c + 1$, loop to step b) until a stop criterion is met, usually a sufficiently good fitness value or G_c is achieve a predefined maximum generation $G_{\max}$.

The parameters of PSO includes: number of particles m, inertia weight w, acceleration constants c_1 and c_2, maximum velocity $V_{\max}$. As evolution goes on, the swarm might undergo an undesired process of diversity loss. Some particles becomes inactively while lost both the global and local search capability in the next generations. For a particle, the loss of global search capability means that it will be only flying within a quite small space, which will be occurs when its location and p_{best} is close to g_{best} (if the g_{best} has not significant change) and its velocity is close to zero for all dimensions; the loss of local search capability means that the possible flying cannot lead perceptible effect on its fitness. From the theory of self-organization [18], if the system is going to be in equilibrium, the evolution process will be stagnated. If g_{best} is located in a local optimum, then the swarm becomes premature convergence as all the particles become inactively.

To stimulate the swarm with sustainable development, the inactive particle should be replaced by a fresh one adaptively so as to keeping the non-linear relations of feedback in Equation (3) efficiently by maintaining the social diversity of swarm. However it is hard to identify the inactive particles, since the local search capability of a particle is highly depended on the specific location in the complex fitness landscape for different problems. Fortunately, the precision requirement for fitness value is more easily to de decided for specified problem. The adaptive PSO is executed by substituting the step d) of standard PSO process, as the pseudo code of adaptive PSO [19] that is shown in Fig. 1. F_i is the fitness of the ith particle, $F_{g\text{best}}$ is the fitness of g_{best}. $\Delta F_i = f(F_i, F_{g\text{best}})$, where $f(x)$ is an error function. The ε is a predefined critical constant according to the precision requirement. T_c is the count constant. The replace ( ) function is employed to replace the ith particle, where the X_i and V_i is reinitialized by following the process in step a) of standard PSO, and its p_{best} is equal to X_i. The array *similar Count* $[i]$ is employed to store the counts which are satisfying the condition $|\Delta F_i| < \varepsilon$ in successively for the ith particle which is not g_{best}. The inactive particle is natural to satisfy the replace condition; however, if the particle is not inactively, it has less chance to be replaced as T_c increases.

```
int[ ]similar Count = new int[m]; // at initializtion stage
// Next code is employed to replace step d)
// in standard PSO process
 For (i = 0; i < m; i + +)  {                    // for each particle
     IF (i ≠ g && |ΔF_i| < ε)
        THEN similar Count[i] + +;               // add1
     ELSE similar Count[i] = 0;                  // reset
     IF (similar Count[i] > T_c)                 // predefinedcount
       THEN replace (the ith particle);
     ELSE execute (step d) in standard PSO
  }
```

Figure 1. Inserted pseudo code of adaptive PSO.

The critical constant ε is set as 0.0001, and the count constant T_c is set as 3. The upper limit of the inertia weight is 0.9 and the lower limit 0.4.

5. GENETICAL SWARM OPTIMIZATION

Some comparisons of the performances of GA and PSO are presented in literature [20], underlining the reliability and convergence speed of both methods, but continuing in keeping them separate. Due to the different search method adopted by the two algorithms, the typical selection-crossover-mutation approach versus the velocity update one,both the algorithms have shown a good performance. In particular PSO seems to have faster convergence rate than GA early in the run, but often it is outperformed by GA for long simulation runs, when the last one finds a better solution. Anyway, the population-based representation of the parameters that characterizes a particular solutionis the same for both the algorithms; therefore it is possible to implement an hybrid technique in order to utilize the qualities and uniqueness of the two algorithms. Some attempts have been done in this direction,

with good results. Most of the times, one technique is used mainly as a pre-optimizer for the initial population of the other technique. In [21], for example, the authors test two different combinations of GA and PSO, using the results of one algorithm as a starting point for the other (in both the orders) to optimize a profiled corrugated horn antenna. Another hybridization strategy is proposed in [22], where the upper-half of the best-performing individuals in a population is regarded as elite and, before using GA operators, it is first enhanced by means of PSO, instead of being reproduced directly to the next generation. The hybrid technique here proposed, called Genetical Swarm Optimization (GSO), and consists in a strong cooperation of GA and PSO, since it maintains the integration of the two techniques for the entire run. In fact, this kind of updating technique yields a particular evolutionary process where individuals not only improve their score for natural selection of the fitness or for good-knowledge sharing, but for both of them at the same time. In each iteration the population is divided into two parts and they are evolved with the two techniques respectively. They are then recombined in the updated population, that is again divided randomly into two parts in the next iteration for another run of genetic or particle swarm operators. Fig. 2 shows the idea that stands behind the algorithm and the way to mixing the two main techniques.

The population update concept can be easily understood thinking that a part of the individuals is substituted by new generated ones by means of GA, while the remaining are the same of the previous generation but moved on the solution space by PSO. The driving parameter of GSO algorithm is the Hybridization Coefficient (HC); it expresses the percentage of population that in each iteration is evolved with GA: so HC = 0 means the procedure is a pure PSO (the whole population is processed according to PSO operators), HC = 1 means pure GA (the whole population is optimized according to GA operators), while $0 < HC < 1$ means that the corresponding percentage of the population is developed by GA, while the rest with PSO technique. For the problem at hand, the HC is equal to 0.5;

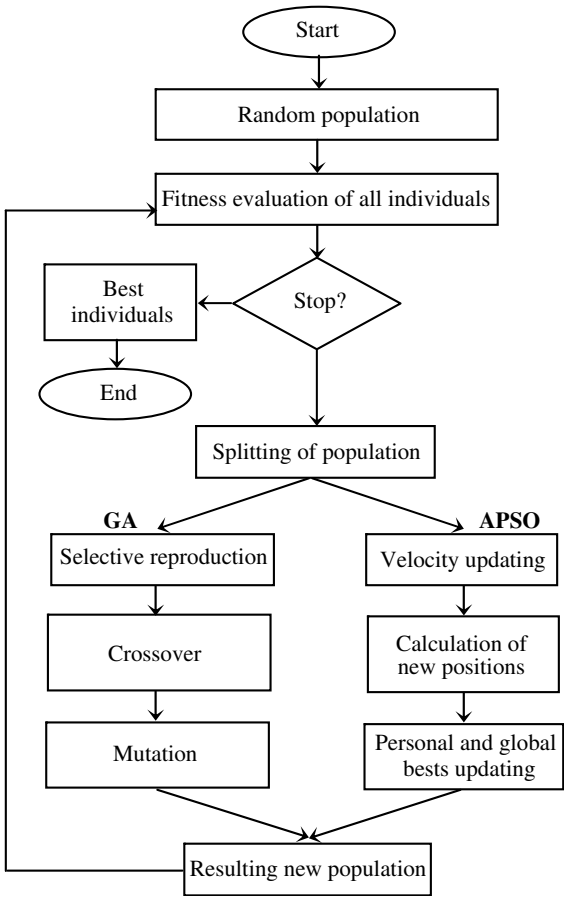

Figure 2. Flow-chart of the GSO algorithm.

the number of dimensions is equal to twice the number of antenna elements because both the amplitude and phase of each parameter must be specified by the GSO. Also, a population of 40 particles was used.

6. PATTERN SYNTHESIS USING GSO

In order to illustrate the capabilities of the hybrid evolutionary algorithm for solving the array configuration for desired pattern synthesis by varying the amplitude and phase of the elements feed, we introduce the case of an array having 10 isotropic elements with $\lambda/2$ spacing, which is supposed to generate two beams steered towards the two angles $-20°$ and $40°$, Fig. 3 shows the output pattern, the relative amplitudes of the two beams were equal to unity, after 297 iterations maximum side lobes level of -25.05 dB was achieved. Amplitude and phase distributions in degree are shown Table 1. the hybrid algorithm is run for 297 generations.

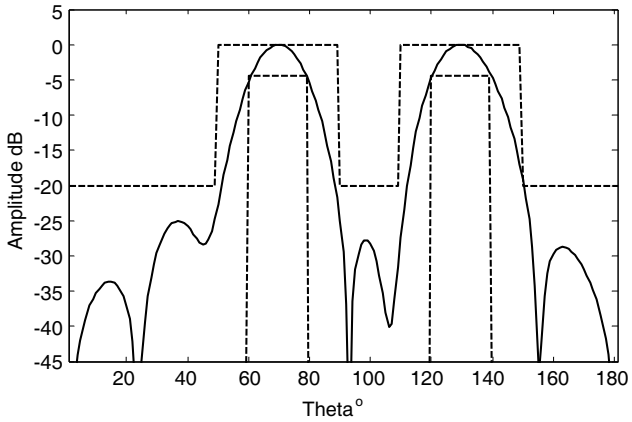

Figure 3. Multi-beam arrays with maximum SLL equal to -25.05 dB.

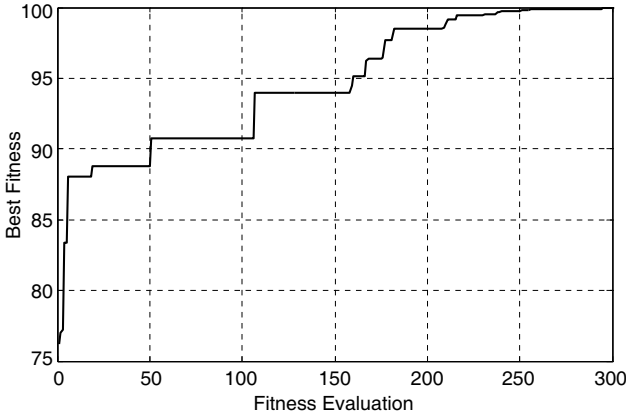

Figure 4. Convergence curve.

Table 1. Amplitudes and phases distributions.

Element No	Magnitude (V)	Phase (°)
1	0.1633	13.2812
2	0.2971	167.332
3	0.4322	−142.064
4	0.6374	52.7236
5	0.7238	79.6010
6	0.7238	−79.6010
7	0.6374	−52.7236
8	0.4322	142.064
9	0.2971	−167.332
10	0.1633	13.2812

We consider an array of 16 isotropic elements spaced 0.5λ apart in order to generate three beams towards the steering angles $-30°$, $10°$ and $50°$ with amplitude-phase synthesis. Because of symmetry, here only eight phases and eight amplitudes are to be optimized. Acceptable side lobe level should be equal to or less than the desired value -20 dB. Fig. 5 shows normalized absolute power pattern in dB the maximum side lobes level reach -20 dB, there is a very good agreement between desired and obtained results. The optimized excitation magnitudes and phases (radian) elements are presented in

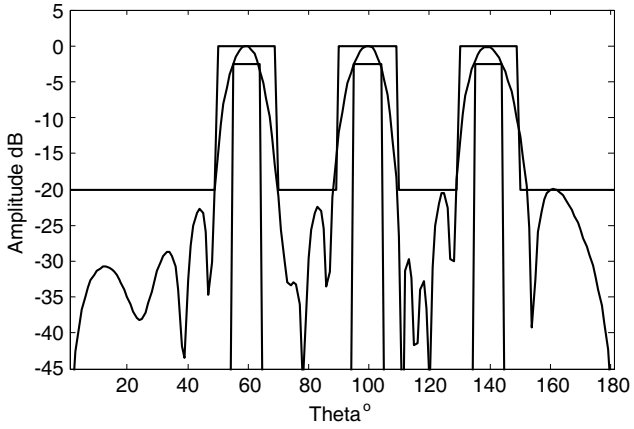

Figure 5. Multi-beam arrays with maximum SLL equal to −20 dB.

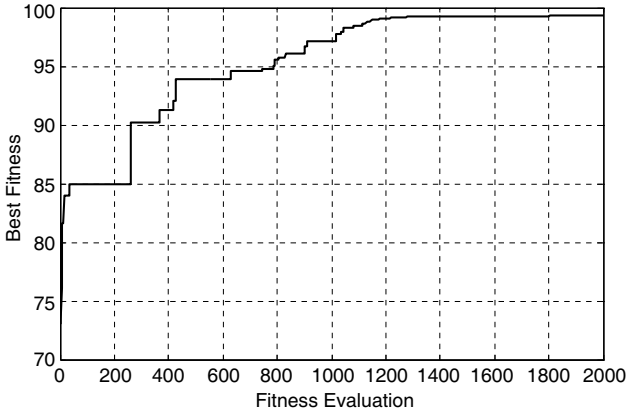

Figure 6. Convergence curve.

Table 2. Amplitudes and phases distributions.

No	Fig. 5		Fig. 7		Fig. 9	
	Ampl (Volt)	**Phase (Rad)**	**Ampl (Volt)**	**Phase (Rad)**	**Ampl (Volt)**	**Phase (Rad)**
1	0.2136	0.780	0.0439	−0.906	0.2382	−2.363
2	0.3599	−2.529	0.1671	−0.339	0.4350	0.578
3	0.1791	−1.699	0.2852	2.071	0.0268	−0.650
4	0.3992	1.838	0.2059	0.499	0.0781	−1.296
5	0.8508	−1.424	0.3031	0.325	0.3626	1.157
6	0.5087	−0.875	0.5421	−1.578	0.8760	−2.004
7	0.3990	2.852	0.6875	0.551	0.7002	−1.833
8	0.8479	−0.172	0.7323	1.440	0.3874	1.644
9	0.8479	0.172	0.7323	−1.440	0.3874	−1.644
10	0.3990	−2.852	0.6875	−0.551	0.7002	1.833
11	0.5087	0.875	0.5421	1.578	0.8760	2.004
12	0.8508	1.424	0.3031	−0.325	0.3626	−1.157
13	0.3992	−1.838	0.2059	−0.499	0.0781	1.296
14	0.1791	1.699	0.2852	−2.071	0.0268	0.650
15	0.3599	2.529	0.1671	0.339	0.4350	−0.578
16	0.2136	−0.780	0.0439	0.906	0.2382	2.363

the Table 2. For design specifications of amplitude-phase synthesis, the hybrid algorithm is run for 1863 generations.

With the same array as the last section and the same type of synthesis, we present synthesis results of multibeam array as indicated in Figs. 7 and 9. Fig. 7 shows normalized absolute power pattern in dB for multibeam array by amplitude-phase synthesis. After 1835 generations, an optimum multibeam pattern of three beams steered towards the three angles −30°, 0° and 40° is obtained. Side lobes level obtained for desired pattern is −20.59 dB. Simulated results are shown in Table 2.

After 1649 iterations, the fitness value reached its maximum, and the optimization process ended due to meeting the design goal. Fig. 9 shows the normalised absolute power pattern for the array with four beams steered towards the four angles −40°, −10°, 20° and 50°. maximum side lobe level reach

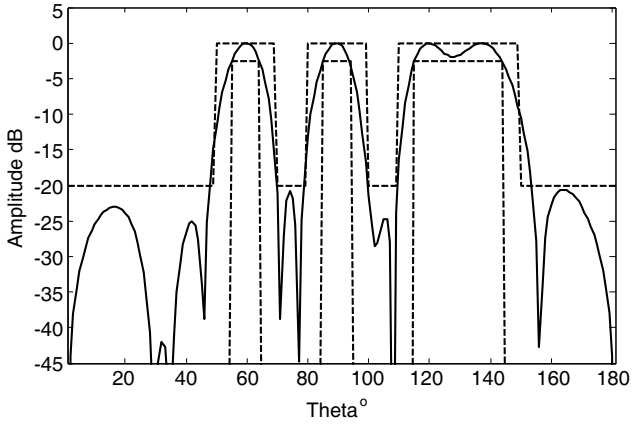

Figure 7. Multi-beam arrays with maximum SLL equal to −20.59 dB.

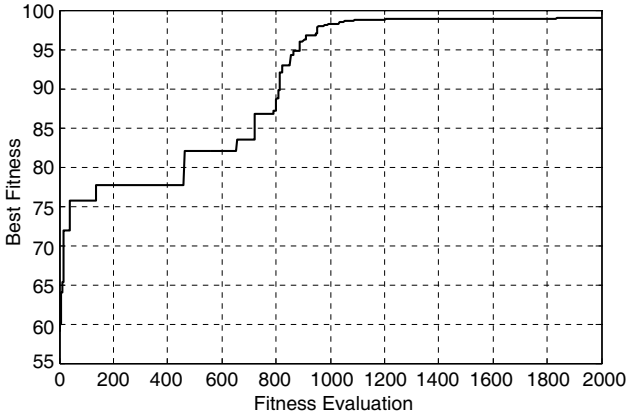

Figure 8. Convergence curve.

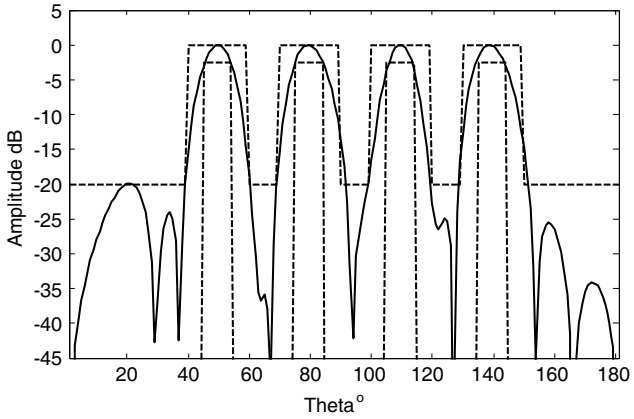

Figure 9. Multi-beam arrays with maximum SLL level equal to −20 dB.

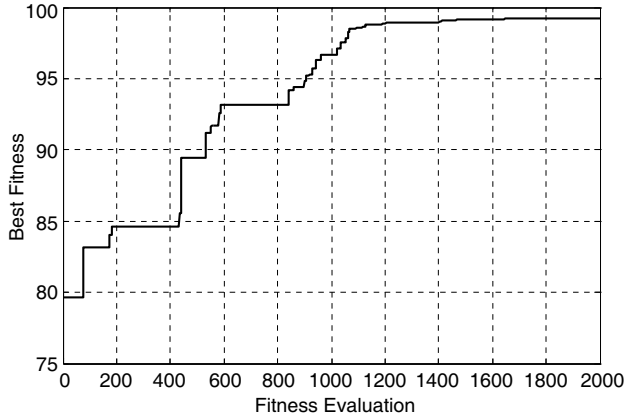

Figure 10. Convergence curve.

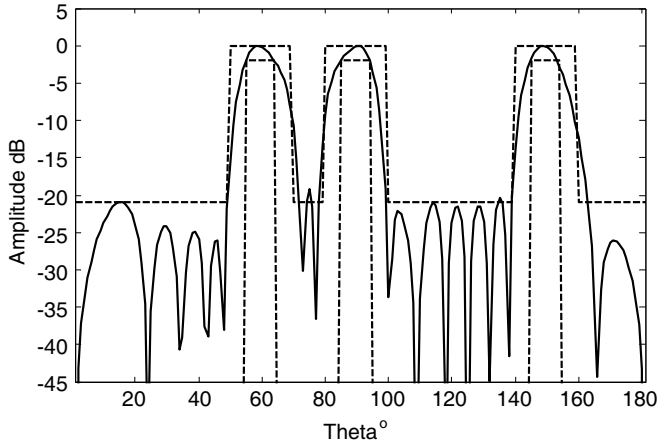

Figure 11. Multi-beam arrays with maximum SLL equal to −20 dB.

−20 dB. The element excitation required to achieve this desired pattern is presented in Table 2. The optimized result indicates that 16 elements symmetrically linear array is capable to realize the design goal with less number of antenna elements.

In order to evaluate the performance of the proposed algorithm, we compare the numerical results calculated by the Hybrid model and the Taylor-Kaiser [1]. We show the comparison of the gains of two 20-element three-beam arrays with half wavelength spacing and steered towards the three angles of −30°, 0° and 60° among the hybrid algorithm results as indicated in Fig. 11, and the Taylor-Kaiser simulated results in [1]. The hybrid algorithm −20 dB and the relative amplitudes of the three beams were equal to unity, and this result remains comparable to the Taylor-Kaiser.

7. CONCLUSION

In this paper, a hybrid algorithm based on the adaptive particle swarm optimizer and standard genetic algorithm has been presented. The proposed technique has been applied to the design of multi-beam linear antennas array, in order to optimize the excitations law of its elements. The reported results show that the genetic swarm optimiser class of procedures is reliable and effective: this feature makes it suitable for wider application in electromagnetic.

REFERENCES

1. Ghayoula, R., M. Traii, and A. Gharsallah, "Application of the neural network to the synthesis of multibeam antennas arrays," *IEEE Transactions on Engineering, Computing and Technology*, Vol. 14, No. 1, 270–273, August 2006.
2. Panduro, M. A. and C. del Rio-Bocio, "Design of beam-forming networks for scannable multi-beam antenna arrays using corps," *Progress In Electromagnetics Research*, Vol. 84, 173–188, 2008.
3. Eiben, A. E. and J. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.
4. Ares-Pena, F. J., J. A. Rodriguez-Gonzalez, E. Villanueva-Lopez, and S. R. Rengarajan, "Genetic algorithms in the design and optimization of antenna array patterns," *IEEE Trans. Antennas Propagation*, Vol. 47, No. 3, 506–510, 1999.
5. Panduro, M. A., C. A. Brizuela, L. I. Balderas, and D. A. Acosta, "A comparison of genetic algorithms, particle swarm optimization and the differential evolution method for the design of scannable circular antenna arrays," *Progress In Electromagnetics Research B*, Vol. 13, 171–186, 2009.
6. Torn, A. and A. Zilinskas, *Global Optimization*, Vol. 350, Springer-Verlag, 1989.
7. Akdagli, A. and K. Guney, "Shaped-beam pattern synthesis of equally and unequally spaced linear antenna arrays using a modified tabu search algorithm," *Microwave and Optical Technology Letters*, Vol. 36, No. 1, 16–20, January 2003.
8. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.
9. Rahmat-Samii, Y. and E. Michielssen, *Electromagnetic Optimization by Genetic Algorithms*, Wiley, 1999.
10. Panduro, M. A. and D. R. Carlos, "Design of beam-forming networks using corps and evolutionary optimization," *International Journal of Electronics and Communications AEUE Elsevier*, Vol. 63, No. 5, 353–365, 2009.
11. Arabas, J., Z. Michalewicz, and J. Mulawka, "GA vs PS — A genetic algorithm with varying population size," *IEEE International Conference on Evolutionary Computation*, 73–78, 1994.
12. Tanese, R., "Distributed genetic algorithm," *International Conference on Genetic Algorithms*, 434–439, 1989.
13. Haupt, R., "Thinned arrays using genetic algorithms," *IEEE Trans. Antennas Propagation*, Vol. 42, No. 7, 993–999, 1994.

14. Panduro, M. A., "Design of coherently radiating structures in a linear array geometry using genetic algorithms," *International Journal of Electronics and Communications AEUE Elsevier*, Vol. 61, No. 8, 515–520, 2007.
15. Mussetta, M., N. Bliznyuk, P. Pirinoli, N. Engheta, and R. E. Zich, "Application of genetic algorithms for optimization of a FSS reflectarray antenna," *The 10th International Conference on Mathematical Methods in Electromagnetic Theory*, 522–524, September 2004.
16. Kennedy, J., "The particle swarm: social adaptation of knowledge," *IEEE International Conference on Evolutionary Computation*, 303–308, April 1997.
17. Eberhart, R. and Y. Shi, "Particle swarm optimization: Developments, applications and resources," *IEEE International Conference on Evolutionary Computation*, 81–86, May 2001.
18. Nicolis, G. and I. Prigogine, *Self-organization in No Equilibrium Systems: From Dissipative Systems to Order through Fluctuations*, John Wiley, 1977.
19. Xie, X., W. Zhang, and Z. Yang, "Adaptive particle swarm optimization on individual level," *International Conference on Signal Processing*, 1215–1218, Beijing, 2002.
20. Boeringer, D. W. and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Trans. Antennas Propagation*, Vol. 52, No. 3, 771–779, March 2004.
21. Robinson, J., S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horn antenna," *IEEE International Symposium on Antennas and Propagation*, Vol. 1, 314–317, June 2002.
22. Juang, C., "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics*, Vol. 34, No. 2, 997–1006, April 2004.