Progress In Electromagnetics Research B, Vol. 58, 47–57, 2014

Parallel Implementation of Hybrid GSA-NM Algorithm for Adaptive Beam-Forming Applications

Korany R. Mahmoud[1, *] and Safwat Hamad[2]

Abstract—Recently researchers have great interest in using multi-core processors for applications requiring intensive parallel computing. In this paper, an approach for the implementation of hybrid parallel Gravitational Search Algorithm (GSA) and Nelder-Mead (NM) algorithm using open Multi-Processing (OPEN-MP) on multi-core processors is proposed for beam-forming applications. The proposed parallel GSA-NM algorithm is used to optimize the complex excitations, amplitudes and phases, of the adaptive array elements to synthesize the array beam-pattern. The array consists of 24-elements uniformly distributed in a circular configuration. To measure the performance of the proposed approach, the results are compared with those obtained using parallel hybrid CFO-NM, and PSO-NM Algorithms.

1. INTRODUCTION

Smart antenna arrays with adaptive beamforming capability are very effective in the suppression of interference and multipath signals to provide interference reduction and improve the capacity, data rates, and performance of wireless mobile communication [1, 2]. Smart antenna array is one of the most important problems to apply the optimization techniques.

Global optimization techniques such as Particle Swarm Optimization (PSO), Central Force Optimization (CFO), and Bacterial Swarm Optimization (BSO) are well known alternatives for global optimization based on a nature-inspired heuristic [3–5]. Extensive experimentations were applied to compare their performance through a number of case studies in sequential mode. PSO showed to have good performance, low computational complexity, few parameters and gives good results. On the other hand, CFO has a higher computational complexity but it gives better results [6]. Recently, Gravitational Search Algorithm (GSA) is considered as a new optimization technique based on the law of gravity and mass interaction [7]. A set of various standard benchmark functions, synthesis of thinned scanned concentric ring array antenna, and a fully digital controlled reconfigurable concentric ring array antenna problems were examined [7–9]. In most cases the GSA provided superior or at least comparable results with PSO and CFO. In [10], the length and width of a rectangular patch antenna has been calculated using GSA optimization technique. The GSA is proposed in [11] for Direction of Arrival (DOA) estimation method based on maximum likelihood (ML) criteria for a Uniform Circular Array (UCA) of 12 elements. The results are compared with those obtained using PSO and multiple signal classification (MUSIC) algorithms in terms of Root Mean Square Error (RMSE). It is found that, the GSA gives better performance in terms of computed final fitness values and computational time. In [12], planar ultra-wide band (UWB) antennas with irregular radiator shapes are designed using GSA and compared with those obtained using CFO algorithm. The comparison between GSA and CFO showed that, the GSA has better performance than the CFO algorithm in terms of computed final fitness values and computational time.

Received 21 October 2013, Accepted 5 January 2014, Scheduled 10 January 2014

* Corresponding author: Korany Ragab Mahmoud (korany@engr.uconn.edu).

[1] Department of Electronics and Communications Engineering, Faculty of Engineering, Helwan University, Cairo, Egypt. [2] Faculty of Computer and Information Sciences, Ain Shams University, Abbassia, Cairo 11566, Egypt.

However, for time-critical applications such as in adaptive beam-forming applications a large processing time is required. Hence, the demand for a parallel solution that accelerates these computations is required. Therefore, a parallel version of optimization algorithms is proposed and implemented either using Compute Unified Device Architecture (CUDA) applied on a Graphics Processing Unit (GPU) [13, 14] or using Open Multi-Processing (Open-MP). In [14], the parallel implementation of CFO and PSO for beamforming application is presented using CUDA applied on a GPU. The comparative study between CFO and PSO showed that; in the sequential mode, CFO algorithm produces more accurate results than the PSO algorithm. However, PSO algorithm takes less time than CFO algorithm. In parallel mode, the PSO produced more accurate results than the CFO algorithm due to problems faced in parallelizing CFO using CUDA [14].

On the other hand, Open-MP is designed to support portable implementation of parallel programs for shared memory multiprocessor architectures [15]. Open-MP is a set of compiler directives and callable runtime library routines that extend FORTRAN, C and C++ to express shared memory parallelism. Recently, many algorithms are implemented in parallel using Open-MP achieving good results in addition to its simplicity in implementation compared to other parallel methods used as CUDA applied on GPU in which the accuracy is reversely proportional with the speed up rate. Open-MP gives a high accuracy results equal to the accuracy of the sequential implementation. In general, it provides an incremental path for parallel conversion of any existing software, as well as targeting at scalability and performance for a complete rewrite or entirely new software [16].

As explained in previous literatures [6, 17–19], better results can be achieved using hybrid method combining two algorithms, global metaheuristics and local search algorithm. The global metaheuristics is used as an efficient algorithm to localize the "best" areas and the local search method is used to refine the results. Nelder-Mead technique was proposed by John Nelder & Roger Mead (1965) as a technique for minimizing an objective function in a many-dimensional space [20]. The method approximates a local optimum of a problem when the objective function varies smoothly and is unimodal. In [6], a hybrid approach involving Central Force Optimization (CFO) and Nelder-Mead (NM) algorithm is proposed for accurate determination of resonant frequency and feed point calculation of rectangular microstrip antenna elements with various dimensions and various substrate thicknesses. It is found that, hybrid CFO-NM algorithm not only decrease the required processing time, but also gives better results compared to stand-alone CFO algorithm. In [17], a hybrid BSO-NM technique is proposed to optimize a bow-tie antenna for 2.45 GHz applications. The BSO-NM algorithm has produced results better than those generated by stand-alone BSO algorithm. In order to emphasize the benefits of using such hybrid algorithms, the hybrid CFO-NM and BSO-NM algorithms are considered to optimize the design of tri-band slotted bow-tie antenna and hexa-band planar inverted-F antenna, respectively, in [18, 19].

This paper presents an approach for hybrid parallel GSA-NM algorithm implementation using Open-MP on multi-core processors for beamforming synthesize. To measure the performance of the proposed approach, the results are compared with those obtained using other parallel hybrid techniques such as CFO-NM, and PSO-NM. The rest of the paper is structured as follows: Section 2 presents the problem formulation. In Section 3, the parallel algorithms implementations are presented. In Section 4, the results are presented and discussed. Finally, Section 5 outlines the conclusions.

2. PROBLEM FORMULATION

In this work, the array structure consists of 24 isotropic elements uniformly distributed in the x-y plane along the perimeter of a circle of radius r as shown in Figure 1. The distance between adjacent elements is $dc = 0.5\lambda$ where λ is the wavelength. For beam-forming synthesis, at each scenario, the feeding of each antenna element (amplitude and phase) needs to be optimized to maximize the main-lobe to certain directions and minimizing in other directions.

The following objective function rewards the antenna array for maximizing the output power toward the desired signal at φ_i and minimizing the total output power in the direction of the interfering signals at φ_j .

$$Fitness = \sum_{i=1}^{L} |AF(\varphi_i)| - \sum_{j=1}^{K} |AF(\varphi_j)| \quad (1)$$

where the constant L represents the number of desired users, and K represents the number of interferers. $AF(\varphi)$ is the array factor that will be maximized or minimized in specific directions using evolutionary algorithms.

$$AF(\theta, \varphi) = \sum_{n=1}^{N} I_n e^{j[\beta * r * \sin(\theta) * \cos(\varphi - pos_n) - \alpha_n]} \tag{2}$$

where I_n and α_n represent the excitation amplitude and phase of the n-th element, respectively. pos_n is the angular position of the n-th element in x-y plane ($\theta = 90°$), r is the radius of the circular array, and β is the phase shift constant ($2\pi/\lambda$).

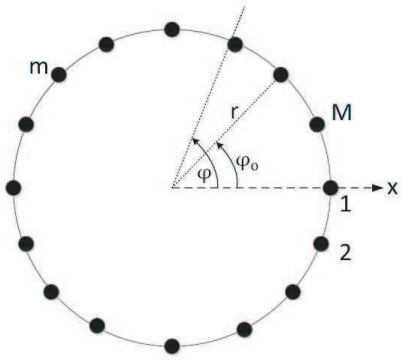

Figure 1. Geometry of the UCA.

3. PARALLEL IMPLEMENTATION OF OPTIMIZATION ALGORITHMS

This section explains parallelism of different optimization algorithms PSO, CFO, and GSA using Open-MP. Firstly, parallel implementation of PSO algorithm will be introduced, and then parallel CFO and GSA will be described.

3.1. Parallel Particle Swarm Optimization (PSO)

The PSO has attracted a lot of attention since its introduction in 1995 [4]. In the PSO, each solution is a point in the search space called a particle in the algorithm. Each particle flies through the D-dimensional problem space learning from the best experiences of all particles. For D-dimensional problem, the position of i-th particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. The particles move to the next position by a rate of position change $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The next position $x_{iD}^{k+1} = x_{iD}^{k} + v_{iD}^{k+1}\Delta t$, where Δt is a unit time step.

The time distribution of PSO in sequential mode (applied on a single thread on CPU) shows that, updating particle's position and acceleration do not takes much time, only 10% of the whole required running time while the step of evaluating fitness takes the remaining 90% of the whole required time. Therefore, the part of evaluating the fitness is the most effective part needs to be parallelized. The parallel implementations of PSO algorithms must have essentially the same structure of the sequential. PSO is easily implemented in parallel using Open-MP applied on a multi-core processor. The following pseudo code shows what is performed for each particles of PSO using Open-MP.

#pragma omp parallel for
For each particle
{
    Update the position of all particles
    Re−evaluate the fitness of all particles
}

For anyone who has ever struggled with threads, making a program multi-threaded with a single line change is amazing! Typically many lines of code must be written, implementing thread pools which dealing with critical sections, semaphores, and mutexes. To run the iterations of the **for-loop** in

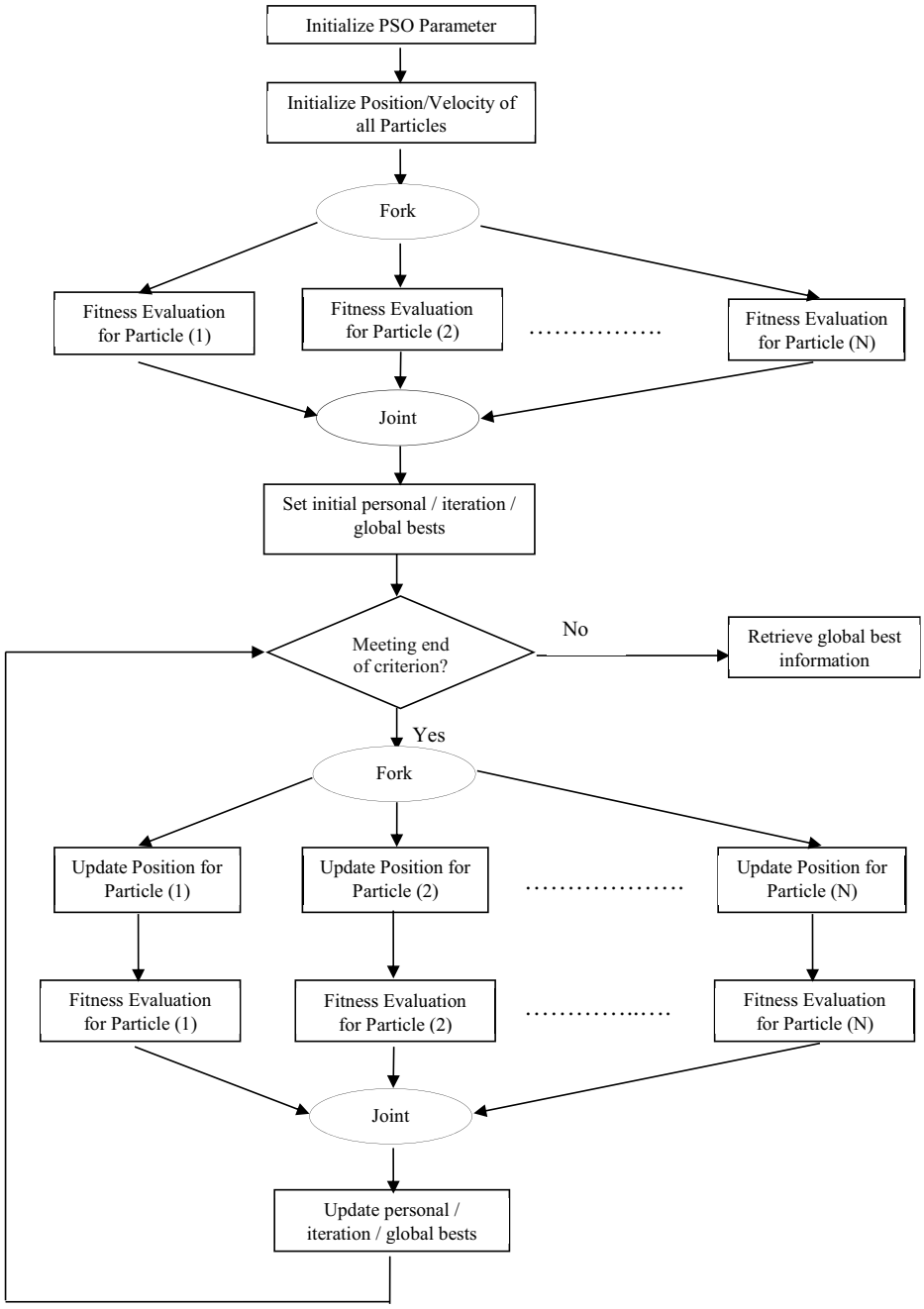

Figure 2. Flow chart of the parallel PSO algorithm.

parallel, the code line **pragma** statement is written before the **for-loop.** Figure 2 shows the flowchart of parallel PSO algorithm.

3.2. Parallel Central Force Optimization (CFO)

Central force optimization (CFO) was introduced as a new deterministic metaheuristics for multi-dimensional search and optimization based on the metaphor of gravitational kinematics [5]. It models "probes" that "fly" through the decision space (DS) by analogy to masses moving under the influence of gravity. Equations are developed for the probe' positions and accelerations using the analogy of probe motion in a gravitational field.

In contrast to PSO, the sequential time distribution of CFO shows that the step of updating the acceleration takes the most processing time of 99.63%. The acceleration update equation in CFO is dependent on the updated position and fitness for all probes. Therefore to be able to implement CFO in parallel, firstly update acceleration for all probes then update position and fitness for all probes. As shown in the pseudo code using Open-MP to make use of multi-threading, some parallelization is added inside the calculation of the acceleration and position. Figure 3 shows the flowchart of parallel CFO

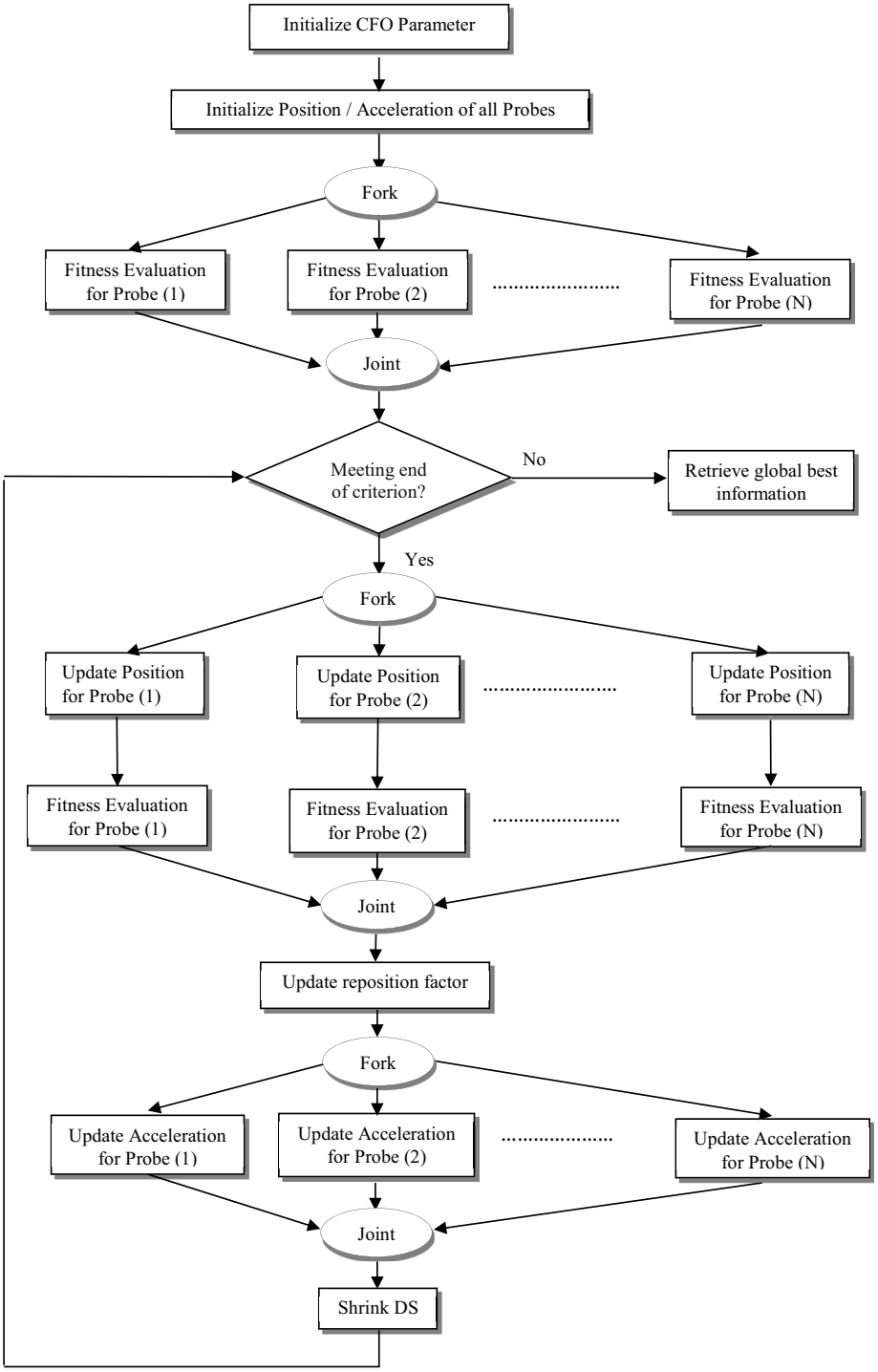

Figure 3. Flow chart of the parallel CFO algorithm.

algorithm.

```
#pragma omp parallel for
For each probe
{
        Update the Acceleration of probe
}
#pragma omp parallel for
For each probe
{
        Update the position of probe
}
#pragma omp parallel for
For each probe
{
        Re-evaluate the fitness of probe
}
```

3.3. Parallel Gravitational Search Algorithm GSA

In GSA, each agent has the following specifications: position and mass. The position of the mass corresponds to a solution of the problem, and its mass is determined using a fitness function. Variation in the velocity or acceleration of any mass is equal to the force acted on the system divided by mass [7]. The sequential time distribution of GSA shows that the step of updating the acceleration takes 79% of the required processing time compared to 99.63% for CFO. GSA is easier to be implemented in parallel with Open-MP compared to CFO. The following pseudo code shows what is performed for each GSA probe.

```
#pragma omp parallel for
For each prob
{
        Update the Acceleration of prob
}
#pragma omp parallel for
For each prob
{
        Update the position and velocity of prob
}
#pragma omp parallel for
For each prob
{
        Re-evaluate the fitness of prob
}
```

The flowchart of parallel GSA algorithm is shown in Figure 4. From experimental it was found that the parts that parallelized and reduced the execution time are"re-evaluate the fitness of particle" and inside the calculation of "update acceleration method". Compared to sequential pseudo code, the **pragma** statement is the only line of code that changed and added before the **for-loop** that needs to be parallelized. This instructs Open-MP to run the iterations of the **for-loop** in parallel.

Unfortunately, the parallelization of Nelder Mead algorithm using Open-MP not affected on the execution time. The main reason is referred to the small for loops in the Nelder Mead so its sequential time is not so much compared to the overhead of the parallelization.

4. RESULTS AND DISCUSSION

To measure the performance of the proposed hybrid parallel GSA-NM algorithm for adaptive beamforming applications, four different scenarios are considered. The tests were performed on HP

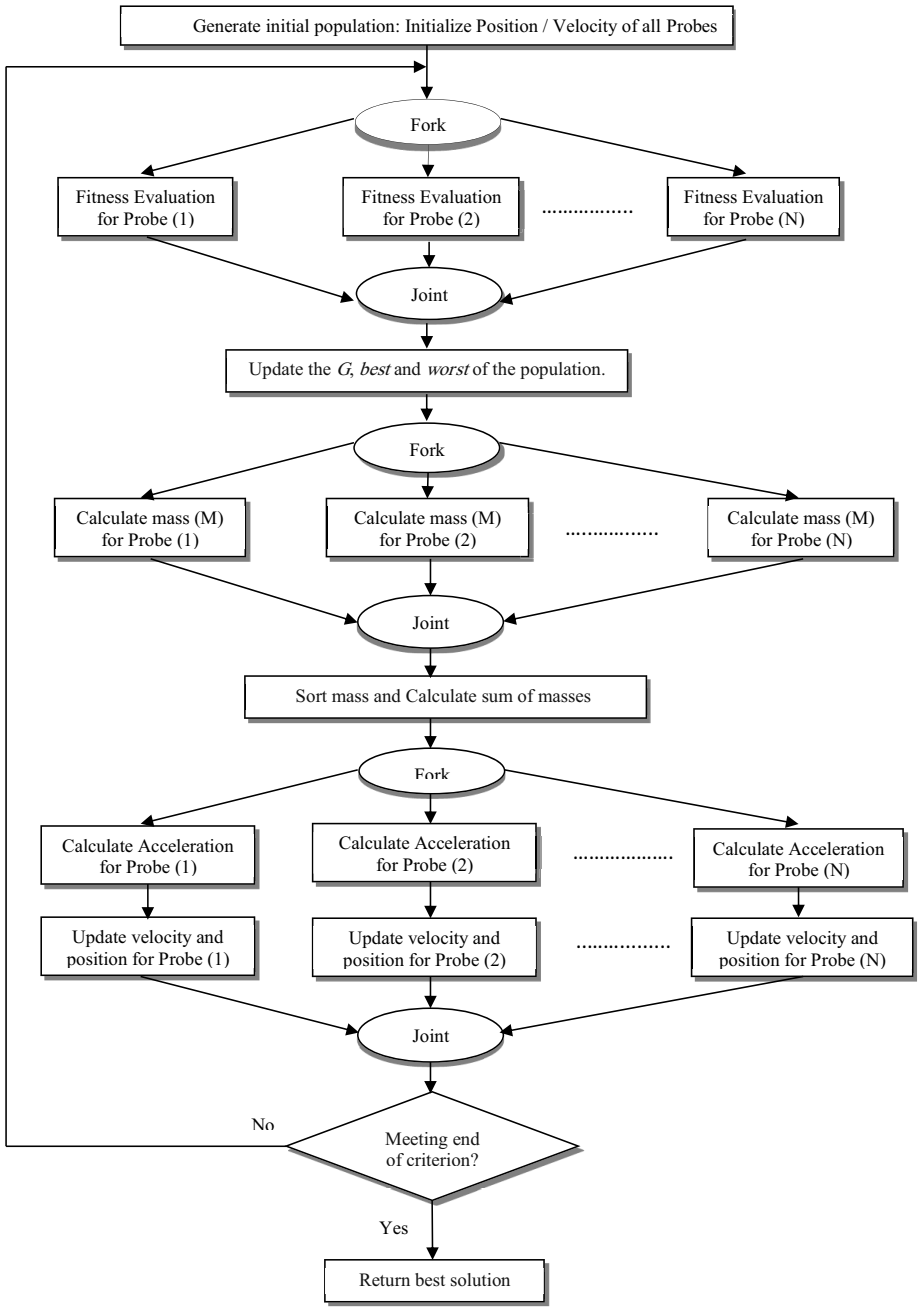

Figure 4. Flow chart of the parallel GSA algorithm.

Pavilion dm4 Notebook PC, Intel® Core(TM) i5 CPU M460 @ 2.53 GHz RAM 4.00 GB run on windows 7 64 bit operating system. Table 1, shows the number of agents and iterations number required for each algorithm either in case of stand-alone or hybrid algorithms in addition to the required processing time.

Firstly, the performance of parallel GSA algorithm in beam-forming synthesize is studied and compared with other parallel PSO and CFO algorithms. As shown in Figure 5 for stand-alone algorithms, the parallel GSA has better performance than parallel CFO and PSO for all considered scenarios which can be clearly identified from the computed final fitness values shown in Figure 6. In average the GSA and CFO outperform the PSO by 65.09%, 29.84% respectively. However, the PSO is found to be faster than CFO and GSA by 5.63, and 3.59 times, respectively. As an advantage of parallel implementation, it is found that the speed up rate of parallel GSA, CFO, and PSO are 1.71, 1.45, and

Table 1. Number of agents and iterations in case of stand-alone and hybrid algorithms.

	STAND-ALONE ALGORITHM			HYBRID ALGORITHM		
	PSO	CFO	GSA	PSO-NM	CFO-NM	GSA-NM
No. of agents for Global optimization	90	180	90	90	180	90
No. of iterations for Global optimization	500	250	500	300	225	450
No. of NM iterations				400	100	100
Sequential (Elapsed CPU time in secs)	**2.86**	**12.41**	**9.35**	**1.71**	**11.2**	**8.38**
Parallel (Elapsed Open-MP time in secs)	**1.52**	**8.56**	**5.46**	**0.86**	**7.53**	**4.86**
Speed up rate	**1.88**	**1.45**	**1.71**	**1.99**	**1.49**	**1.72**

1.88, respectively. In terms of required computational time, it is found that the required processing time for parallel GSA is less than that required for parallel CFO by 36.2%.

Now, the hybrid parallel implementation of GSA-NM, CFO-NM, and PSO-NM algorithms are studied and compared with each other and with parallel stand-alone results. Firstly as shown in Table 1, for parallel hybrid algorithms the number of iterations of global optimizations GSA, CFO, and PSO are deducted to 450, 225, and 300 iterations, respectively, based on the convergence behavior shown in Figure 6. Then Nelder-Mead algorithm is followed with iterations number of 100, 100, and 400 for GSA, CFO, and PSO, respectively to keep the overall evaluation number approximately the same.

As shown in Figure 5, the capability of the uniform circular array (UCA) array to maximize the power in certain directions and minimize in others are significantly improved using parallel hybrid

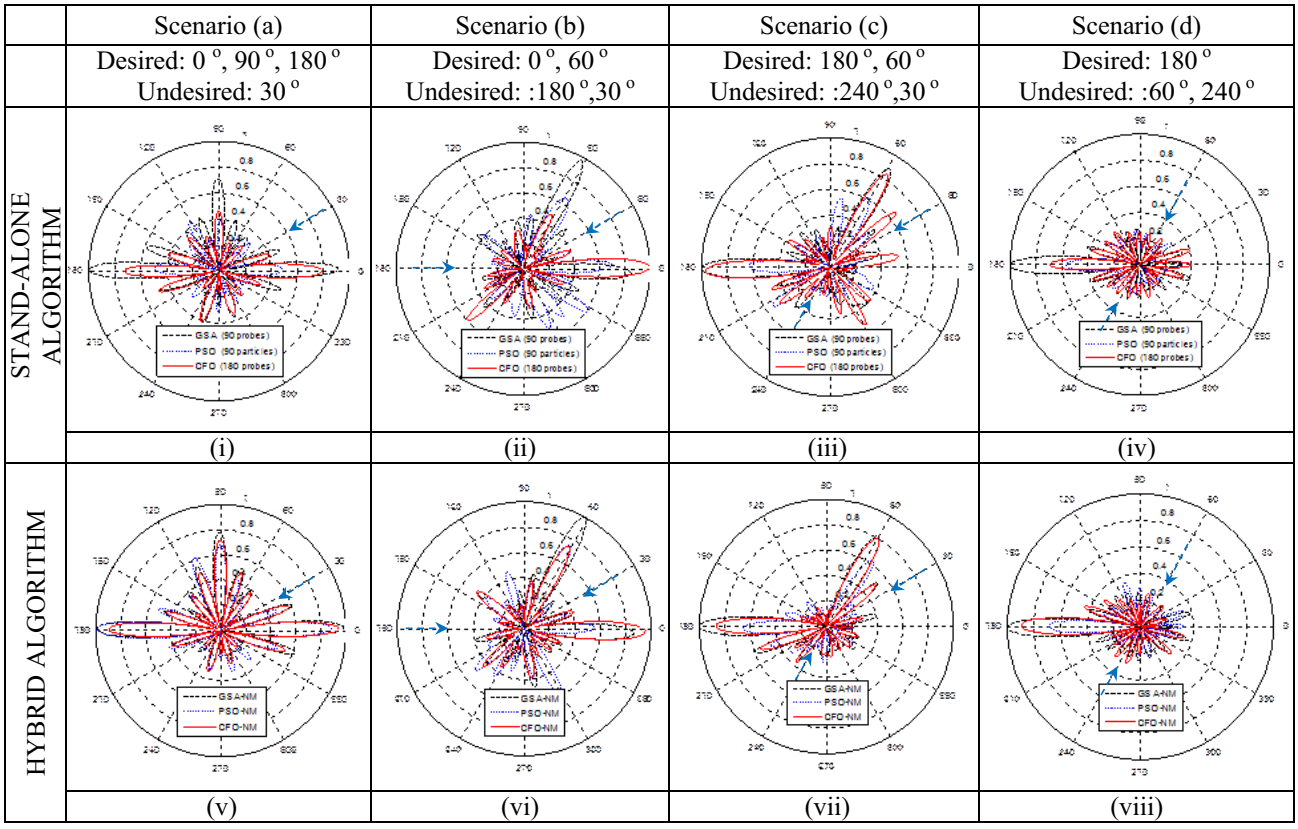

Figure 5. Beam-forming patterns.

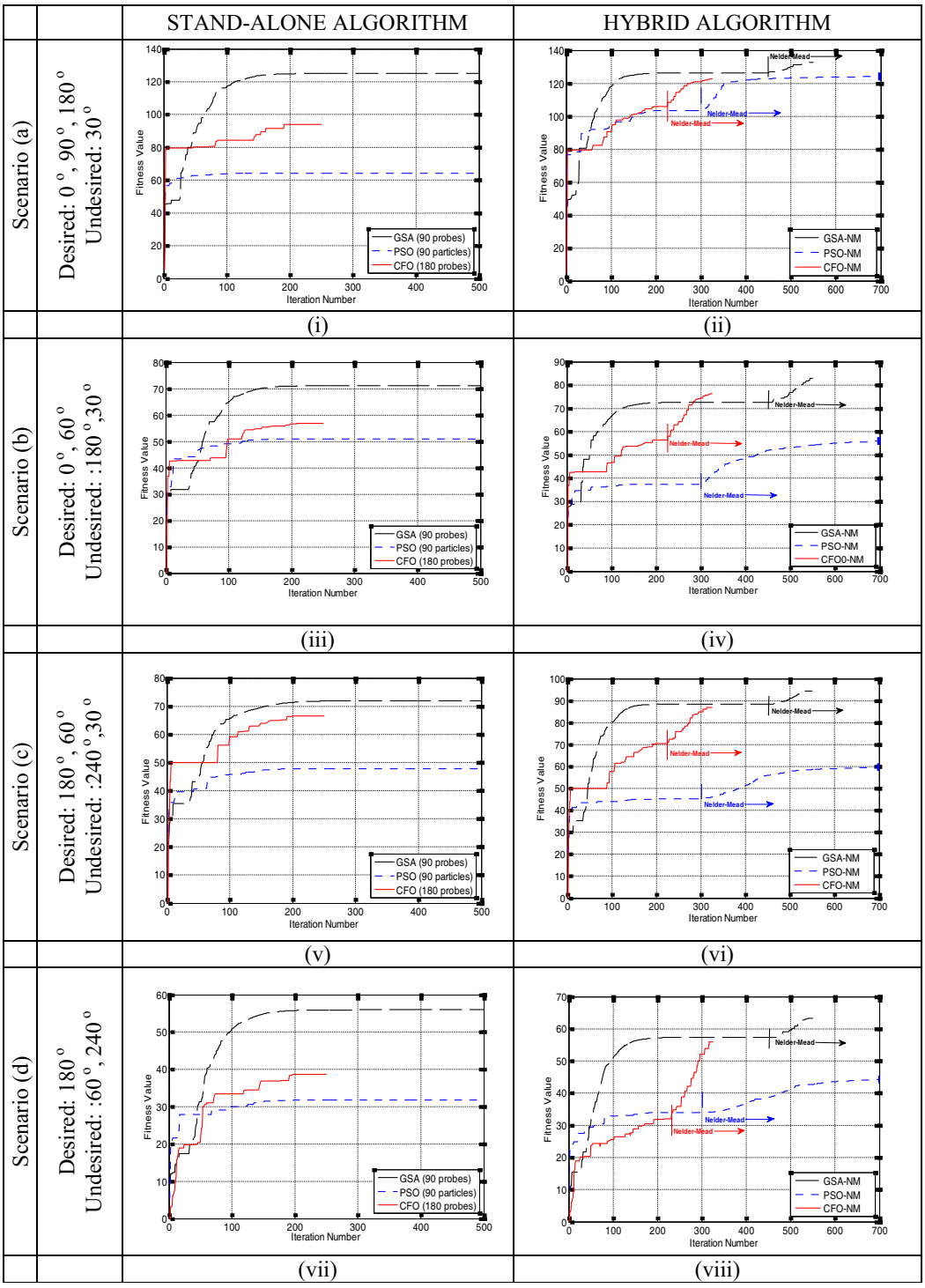

Figure 6. Fitness value versus iterations.

techniques compared to stand-alone algorithms for all considered scenarios. In average, the hybrid GSA-NM, CFO-NM, and PSO-NM algorithms outperforms the stand-alone algorithms by 16.75%, 34.92%, and 41.93%, respectively. In addition the average required processing time is decreased by 10.98%, 12.03%, and 43.42% for GSA, CFO, and PSO respectively.

As shown in Figure 7, the sequential hybrid GSA-NM algorithm is found to be faster than CFO-

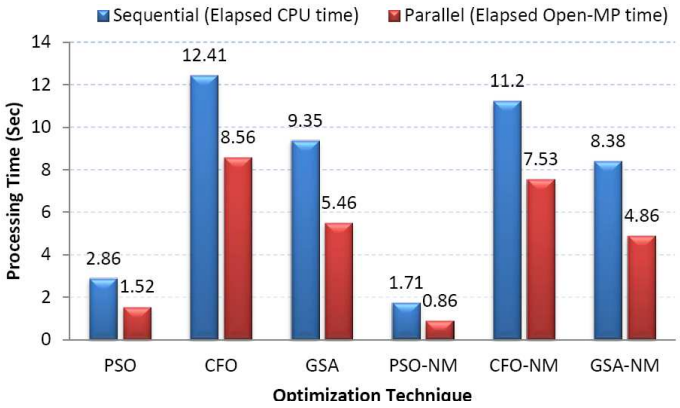

Figure 7. The processing time for sequential and parallel algorithms.

NM algorithm by 25.17%. While the processing time of parallel GSA-NM using Open-MP is found to be faster than parallel CFO-NM algorithm by 35.46% due to the higher speed up rate for GSA-NM algorithm which is 1.72 compared to 1.49 for CFO-NM algorithm.

Comparing the required processing time of parallel hybrid PSO-NM algorithm to CFO-NM and GSA-NM, it is found that the speed up rates is increased to 8.75, 5.65, respectively. Generally, the PSO is found to be faster than both algorithms due to algorithm simplicity, but with worse performance. It should be declared that, for the complexity analysis of the considered parallel algorithm it is found that, Big.O for the PSO algorithm is $O(N^2)$. However, Big.O for the both GSA and CFO algorithm is found to be $O(N^3)$ where N is the maximum number of particles.

Although GSA has the same idea of CFO, it performs different computations and hence outperforms the CFO optimization technique for adaptive antenna beam-forming.

5. CONCLUSION

In this paper, a parallel GSA is implemented using Open-MP to accelerate the computations required for beam-forming synthesize. Then the results are compared with those obtained using PSO and CFO algorithms. The comparison shows how the parallel version of the PSO, CFO and GSA outperforms the sequential one in terms of reducing time by 46.85%, 31.02%, and 41.6%, respectively. Also, the parallel hybrid GSA-NM algorithm is proposed and compared to CFO-NM, PSO-NM algorithms. It is found that the hybrid algorithms outperformed stand-alone algorithms by 16.75%, 34.92%, and 41.93% in addition to decreasing the average required processing time by 10.98%, 12.03%, and 43.42% for GSA, CFO, and PSO, respectively.

REFERENCES

1. Lehne, P. H. and M. Pettersen, "An overview of smart antenna technology for mobile communications systems," *IEEE Commun. Surveys Tutorials*, Vol. 2, 2–13, 1999.
2. Chryssomallis, M., "Smart antennas," *IEEE Antennas Propag. Mag.*, Vol. 42, 129–136, 2000.
3. Mahmoud, K. R., M. El-Adawy, R. Bansal, S. H. Zainud-Deen, and S. M. M. Ibrahem, "Analysis of uniform circular arrays for adaptive beamforming applications using particle swarm optimization algorithm," *Int. J. of RF and Microwave Computed Aided Eng.*, Vol. 18, 42–52, 2008.
4. Kennedy, J. and R. Eberhart, "Particle swarm optimization," *IEEE International Conference on Neural Networks*, Vol. 4, 194–1948, Perth, Australia, 1995.
5. Formato, R. A., "Central force optimization: A new metaheuristic with applications in applied electromagnetics," *Progress In Electromagnetics Research*, Vol. 77, 425–491, 2007.
6. Mahmoud, K. R., "Central force optimization: Nelder-Mead hybrid algorithm for rectangular microstrip antenna design," *Electromagnetics*, Vol. 31, 578–592, 2011.

7. Rashedi, E., H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, Vol. 179, No. 13, 2232–2248, 2009.
8. Chatterjee, A., G. K. Mahanti, and N. N. Pathak, "Comparative performance of gravitational search algorithm and modified particle swarm optimization algorithm for synthesis of thinned scanned concentric ring array antenna," *Progress In Electromagnetics Research B*, Vol. 25, 331–348, 2010.
9. Chatterjee, A., G. K. Mahanti, and P. R. S. Mahapatra, "Design of fully digital controlled reconfigurable dual-beam concentric ring array antenna using gravitational search algorithm," *Progress In Electromagnetics Research C*, Vol. 18, 59–72, 2011.
10. Altinoz, O. T. and A. E. Yilmaz, "Calculation of optimized parameters of rectangular patch antenna using gravitational search algorithm," *2011 International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 349–353, June 15–18, 2011.
11. Magdy Mohamed, A., K. R. Mahmoud, S. G. Abdel-Gawad, and I. I. Ibrahim, "Direction of arrival estimation based on maximum likelihood criteria using gravitational search algorithm," *PIERS Proceedings*, 1162–1167, Taipei, March 25–28, 2013.
12. Mahmoud, K. R., "UWB antenna using gravitational search algorithm," *Journal of Engineering Sciences*, Vol. 41, No. 5, Faculty of Engineering, Assiut University, September 2013.
13. Green, R., L. Wang, M. Alam, and R. A. Formato, "Central force optimization on a GPU: A case study in high performance metaheuristics," *Journal of Supercomputing*, Vol. 62, 378–398, October 2012.
14. Ahmed Fahmy, E., K. R. Mahmoud, S. H. Hamad, and Z. T. Fayed, "Real time parallel PSO and CFO for adaptive beam-forming applications," *PIERS Proceedings*, 816–820, Taipei, March 25–28, 2013.
15. Jin, H., M. Frumkin, and J. Yan, "The Open-MP implementation of NAS parallel benchmarks and its performance," *MRJ Technology Solutions*, NASA Contract NAS2-14303, Moffett Field, CA 94035-1000, October 1999.
16. Candan, C., J. Dréo, P. Savéant, and V. Vidal, "Parallel divide-and-evolve: Experiments with Open-MP on a multicore machine," *GECCO*, 1571–1578, ACM, 2011.
17. Mahmoud, K. R., "Design optimization of a bow-tie antenna for 2.45 GHz RFID readers using a hybrid BSO-NM algorithm," *Progress In Electromagnetics Research*, Vol. 100, 105–117, 2010.
18. Montaser, A. M., K. R. Mahmoud, and H. A. Elmikati, "Tri-band slotted bow-tie antenna design for RFID reader using hybrid CFO-NM algorithm," *29th Nationa Radio Science Conference (NRSC 2012)*, 119–126, April 10–12, 2012.
19. Mahmoud, K. R., A. M. Montaser, and H. A. Elmikati, "Design of Hexa-band planar inverted-F antenna using hybrid BSO-NM algorithm for mobile phone communications," *Int. J. of RF and Microwave Computed Aided Eng.*, Vol. 23, No. 1, 99–110, 2013.
20. Nelder, J. A. and R. Mead, "A simplex method for function minimization," *Computer Journal*, Vol. 7, 308–313, 1965.