World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 8, No. 6, 43-49, 2018

Broadcast Distributed PSO for Large Scale Problems

Farid Bourennani
Department of Computer Science
Faculty of Computing and IT, University of Jeddah, Saudi Arabia

Abstract— Nowadays, we have access to unprecedented high-performance computing (HPC) resources that can be utilized to solve complex and computationally expensive optimization problem. However, one of the problems with existing metaheuristics algorithms is that they do not scale well. For example, particle swarm optimization (PSO) which is one of the most known metaheuristics performs poorly in terms of accuracy and convergence speed with large dimensional problems. In this paper, we propose a broadcast and distributed PSO using message passing interface (MPI) that showed to be faster and more accurate than the commonly utilized distributed master-slave version of PSO for the studied large-scale optimization problems.

Keywords- Particle swarm optimization; distributed computing; large scale optimization; big optimization.

I. INTRODUCTION

Various optimization problems are difficult to solve using exact optimization methods [1] such as software engineering, energy systems design, bioinformatics, telecommunication, and finance among others because they encompass complexities such as discontinuity, large dimensionality, non-differentiability, non-concavity, multimodality, and/or black box problems [2][3]. Consequently, bio-inspired methods such as particles swarms and evolutionary algorithms are the main alternatives to solve such complex problems [4][5]. These nature-inspired computational methods tend to optimize a problem by iteratively trying to improve a candidate solution; they do not guarantee optimality but in general they do provide optimal or close to optimal solutions [6]. A comprehensive description of such methods referred as metaheuristics can be found in [7].

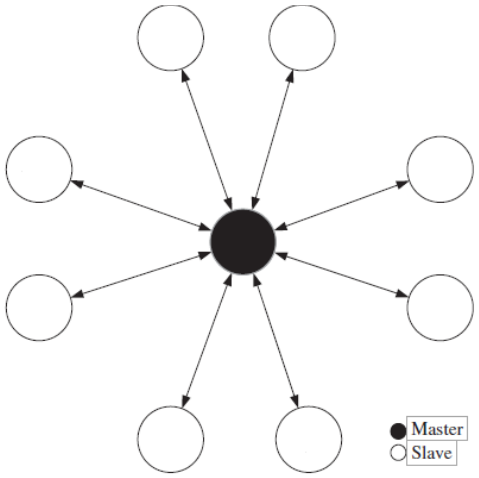

Figure 1: Master-Slave PSO. Every circle represents a particle that communicates with leader (master) particle at every iteration.

The fast hardware development invigorated the development of advanced simulation tools that are computationally intensive due to expensive evaluation functions [8] such as design of energy systems [9] or mechanical engineering models [10] which can involve a large number of variables. Therefore, high performance computing (HPC) arises naturally to solve such problems. Particle swarm optimization (PSO) [11] is the most popular metaheuristic due to its efficiency, there have been over 1000 publications per year related to PSO since 2010 [12]. However, despite the advantage of high-performance computing, PSO does not scale well with large dimensional problems [12], [13]. For example, the master slave version of PSO (MS-PSO), illustrated in Figure 1, is the most known version of distributed PSO, but it suffers from bottlenecks which are due to centralized processing at the master particle level at every iteration because of the data provided by the entire swarm to the master particle. This is especially true with large dimensional optimization problems where the number of variables is high in consequence of which the size of the population is also increased.

To solve this problem, in this work, we propose a broadcast distributed model of PSO (BD-PSO) where there is no master node, rather the entire particles do identical tasks which eliminates the bottleneck caused by a centralized communication at the master node in consequence of which the convergence speed is improved due to asynchronous communication. Furthermore, when solving large dimensional optimization problems (D>100), a higher accuracy is achieved because 1) the proposed BD-PSO enhances the pseudo-randomness [14] when generating new solutions on various nodes rather than having the same master node generating all new solutions, and 2) the best found solution is broadcasted faster to the swarms due to the asynchronous communication model. For testing purposes, we utilize the CEC2013 competition [9] benchmark problems which can take several hours to be solved using a personal computer. The distribution

of PSO is done using the MPI library. We compare the BD-PSO with the classical PSO in terms of accuracy, and we compare it to the MS-PSO in terms of convergence speed in order to demonstrate that the use of a decentralized and asynchronous approach can lead to faster and more accurate results especially when solving large scale optimization problems.

II. LITERATURE REVIEW

Development of distributed metaheuristics is a very active research area; a comprehensive survey about distribution of metaheuristics can be found in [15]. However, as we focus in this work on distributed PSO (DPSO), we will present in this section only the most important works related to DPSO.

Most works used distribution or parallelisation for the sole purpose of accelerating the convergence speed of PSO using the master/slave (M/S) model. For example, in [16] the M/S model was utilized to solve a job shop problem with blocking. The proposed algorithm was compared with a distributed and a classical branch and bound (BB) using four nodes. The results of the proposed DPSO were either similar or more accurate than the sequential PSO depending on the problem; however, the BB achieved better results in most of the instances. In [17], a DPSO was proposed with M/S model using C-Cuda. Most of the code is executed is parallel; five benchmark problems were utilized for comparison purposes. Only the acceleration was studied; the results showed respective speedups of [4.44, 16.9] and [4.5, 17.3] for 10,000 and 100,000 iterations. In [18], the M/S DPSO model was utilized to solve a coverage problem of pursuit-evasion games using MPI with respectively 1, 2, 4, 5 and 10 processors, and a population size of 20. The results showed a speedup of over 3.0 with 4 processors which decreased to over 2.5 with 10 processors. Also, there are several applied works which utilized MS-based DPSO. For example, in [19] MS-DPSO was utilized with GPU for the optimization of an antenna adaptive beam forming problem. In [20], a DPSO named PSO-LSSVM was proposed with a M/S communication model using Hadoop (3 processors on a cloud) to predict the benefits of electronic commerce over a period of two years. The results of PSO-LSSVM were the most accurate only during the second-year period. In [21], MS-DPSO was proposed using map reduce paradigm to solve an intrusion detection problem. The speed-up was only linear until to eight nodes. In [22], MS-DPSO was utilized to optimize the soil sampling network in China. In addition to speed, the proposed DPSO required only 85 % of the samples needed compared to classical PSO.

Some other works proposed other DPSO but focused only on low dimensional problems (D<30) such as [23], [12] and [24]. Some other works proposed DPSO for especial applications such as robotics [25-28] or multi-agent platform [29]. There are also multi-objective version of DPSO that can be found in [30], [31], and [32].

An interesting work was proposed by [32] in parallelizing COMPSO which is a cooperative version of PSO. Two different distributed platforms were used, namely, Sharcnet and a shared-memory multi-core PCs. Five problems were utilized for testing purposes with a respective number of variables of 300, 600, and 1000. The subswarms populations were composed of five particles for a total abnormally large size population [33-34] which are respectively composed of 500, 1000, and 2000 particles. Up to 33 nodes were utilized. Interestingly, only low numbers of CPU (1, 3, and 9) achieved the most accurate results compared to 17, 25, and 33 CPUs. However, in this case every particle works only on a part of the solutions, which means only a portion of variables is modified to generate new solutions which differs from the classical way PSO works. Although, the paper is rich of experiments, they don't compare the proposed algorithms with classical PSO in terms of accuracy.

Overall, it can be said that despite the high popularity of PSO, limited work has been done of PSO to attempt to solve large scale optimization problems using high performance computing which ultimately could lead to solve many real-life problems. In addition, the use of HPC can improve the randomness of variables by distributing them across heterogeneous cores (CPUs) [14]. Also, the use of distributed nodes, which can be of very large numbers, will permit to solve large scale optimization problems which cannot be solved with shared-memory platforms due to limited memory capacity [8]. Furthermore, asynchronous communication models are faster because they reduce unnecessary communication and bottlenecks caused by synchronization [35] [23] in consequence of which the population tend to move more promptly [25].

III. BROADCAST DISTRIBUTED PSO

In this section, we describe the proposed algorithm called broadcast and distributed PSO (BD-PSO). The broadcast communication model serves to eliminate bottlenecks caused by centralized communication at the master particle and have instead an asynchronous broadcast communication model among all particles to share the best-found solution information among the swarm as soon as it is found which will lead to a faster convergence speed. The implementation of BD-PSO is done on the shared hierarchical academic research computing network (SHARCNET) which is an academic cluster for distributed computations platform which is based on MPI.

By proposing BD-PSO, we target the enhancement of both the accuracy and convergence speed when solving large scale optimization problems that would be computationally too expensive to be solved using classical computation. BD-PSO uses classical parameter settings with a constant population size as recommended in [37] or based on the number of variables of a problem n using the rule $2n < P < 5n$ [35]. Recent works [34] recommended even to have population sizes significantly smaller than the dimensions $P \ll n$. To assess PB-PSO in terms of accuracy and acceleration, BD-PSO is compared with both PSO and MS-PSO which is the most utilized DPSO.

Algorithm 1: BD- PSO
Code executed at every node rather than Master node.

Random initialization of whole swarm
repeat
for all particles *i* **do**
Update velocities:
$\vec{v}_{\mathrm{i}}(t) = w.\vec{v}_{\mathrm{i}}(t-1) + C_1.r_1(\vec{x}_{p_i} - \vec{x}_{\mathrm{i}}) + C_2.r_2(\vec{x}_{G_i} - \vec{x}_{\mathrm{i}})$ (t)
Move to the new position: $\vec{x}_{\mathrm{i}}(\mathrm{t}) = \vec{x}_{\mathrm{i}}(t-1) + \vec{v}_{\mathrm{i}}(t)$
if $f(\vec{x}_{\mathrm{i}}) < f(\vec{x}_{p_i})$**then**
$\vec{x}_{p_i} = \vec{x}_{\mathrm{i}}$
end if
if $f(\vec{x}_{\mathrm{i}}) < f(\vec{x}_{G_i})$**then**
$\vec{x}_{G_i} = \vec{x}_{\mathrm{i}}$
InformOtherParticles($\vec{x}_{G_i}$);
end if
Update($\vec{x}_{\mathrm{i}}$, $\vec{v}_{\mathrm{i}}$)
end for
until Stopping Criteria

where $\vec{x}_{p_i}$ is the (local) best found candidate solution by the particle $\vec{x}_{\mathrm{i}}$, $\vec{x}_{G_i}$ is the (global) best found particle in the entire swarm called leader. w is the particle inertia weight representing a trade-off between the global and local experiences, r_1 and r_2 are random variables in the range [0,1], and C_1 and C_2 are learning factors towards respectively the particles personal success and its neighbor's success.

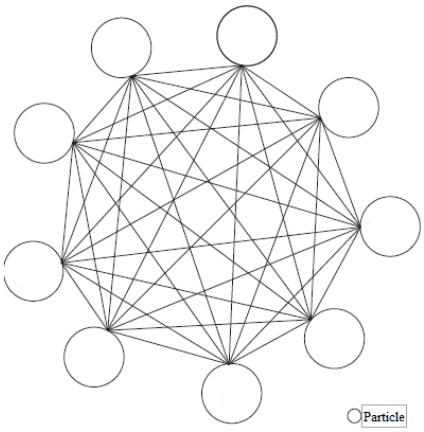

Figure 2. Broadcast communication model among PSO particles.

The proposed BD-PSO will consists in eliminating the master particle by distributing equally all the tasks to the entire particles of the swarm. Consequently, all the particles will have the same tasks; however, they will share their found solution at every iteration. This way, there will be no idle processing node, as in the master-slave model, while the master particle located on the node does the comparison of the found solution. The number of nodes is equal to the number of particles. As shown in Figure 2, the communication process is more intense as compared the master-slave model because of the broadcasting done at every iteration. However, the butterfly effect of the "AllReduce()" function will alleviate that effect. Also, as shown in Algorithm 1, the function "InformOtherParticles()" will serve to inform the other particles of the new best found solution as soon as it is found using asynchronous communication with the aim of achieving a faster convergence speed and also a better accuracy. It is important to mention that random generation function of every particle located on a separate node is completely independent, i.e. the pseudo-randomness effect is reduced which also should lead to a better accuracy.

IV. Experiments Settings

In this section, we present the experiments details which include parameter settings, benchmark problems description, measuring metrics for assessment performance, and the experiments results.

A. Parameter Settings

For a fair comparison, we kept identical parameters settings for all the algorithms as suggested in [22]: population size = 100, w = 0.729844, number of function calls =1*106 per swarm, C1 = C2 = 1.49618, and the number of nodes = 100 for distributed PSOs.

B. Benchmark Optimization Problems

Eight well-known scalable benchmark problems [9], that are commonly utilized, have been used for testing purposes; they are composed of different landscape properties such as multi-modality, deceptiveness, and non-convexity. The global optima of these problems are known beforehand.

To make the selected problems complex to solve, every problem is set respectively to 100 then to 500 variables which makes them time consuming (several hours) to be solved sequentially. In addition, every problem is shifted to a random location in a search space which makes the problem further harder to solve.

C. Comparative Metrics

The most common metrics utilized by the metaheuristics research community to assess the performance of distributed algorithms are the speedup (Ψ) and accuracy (α) [38].

The speedup measures how fast is a distributed algorithm in comparison with its corresponding sequential version. It is calculated as the ratio of the sequential execution time over the parallel execution time. The ideal case scenario for a distributed algorithm should lead to a linear speedup whereas in most cases a sub-linear speedup is achieved. The poor (sub-linear) speedup is due to either the serial portion of the code or the distributed overhead. The speedup is described below.

$$\Psi(n,p) = T\frac{n,1}{T}(n,p) \qquad (4)$$

where n is the population size, p is the number of processors, $T(n,1)$ is the time required to solve a problem of size n on a single processor (serial processing), and $T(n,p)$ is the time required to solve a problem of size n on p processors (distributed processing).

The other measure used for comparison is the accuracy (α) of an algorithm which consists in calculating the difference between the fitness of the best-found solution and the global optimum as shown below. The closer to zero is the value of α, the more accurate is the optimization algorithm. An ideal optimization algorithm should have α equal to zero.

$$\alpha = \left| S_f(x) - \hat{S}_{f(x)} \right| \qquad (6)$$

where $S_f(x)$ is the best-found solution by the algorithm and $\hat{S}_{f(x)}$ is the global optimum of the function $f(x)$.

For a fear accuracy comparison, the stopping criteria is fixed for all the compared PSO methods to 1*106 function calls which is classically utilized [9].

V. Experiments Results

In this section, we present the experimental results by comparing the proposed BD-PSO with the parent PSO algorithm and MS-PSO. Two series of tests are performed for the evaluation of the performance of the algorithms with large dimensional problems which are respectively 100 and 500 variables.

A. Experiments with 100 variables

Both the Figure 3 and Table 1 show the execution time of the three compared algorithms when solving problems of dimension D = 100. It can be seen that the proposed BD-PSO is the fastest algorithm for all the problems which is due to the broadcast communication model.

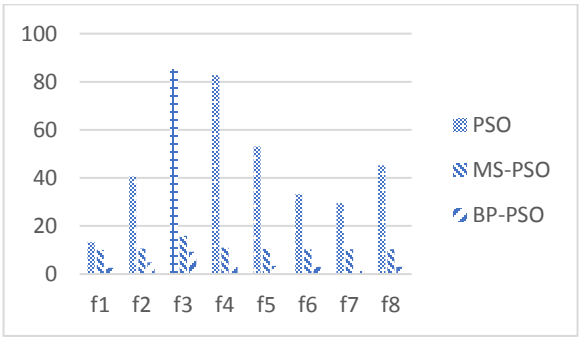

Figure 3. Execution time in seconds with functions scaled to 100 variables. The execution time of PSO for f3 has been shortened for better visualization. Shorter histograms are of better results.

Table 2 and 4 show the speed-up when solving the eight benchmarking problems set to 100 variables after 1,000,000 function calls. BD-PSO achieved the highest speedup with an average of 12.0 compared to 4.2 for MS-PSO which makes BD-PSO in average three time faster than MS-PSO. This performance of BD-PSO is due to the elimination of bottleneck communication at the master particle which is still persistent within the MS-PSO algorithm. For the most expensive problem f3 the speedup of PB-PSO is the closest to MS-PSO with BD-PSO being 1.7 faster than MS-PSO, and for the less expensive problem f1 the BD-PSO achieved the highest speed-up difference with BD-PSO being 3.9 faster than MS-PSO. So, the less expensive is the objective function, the faster is the convergence speed of an algorithm using a broadcast communication model.

TABLE I. Execution time in seconds with functions scaled to 100 variables (Dark grey is the fastest algorithm; light grey is the second fastest algorithm)

Function	PSO	MS-PSO	BD-PSO
f1	13.3	10	2.6
f2	40.6	10.5	4.9
f3	386.2	15.9	9.3
f4	82.7	11	3.9
f5	53	10.4	3.4
f6	33.3	10.3	3.0
f7	29.5	10.4	2.9
f8	45.1	10.3	2.9

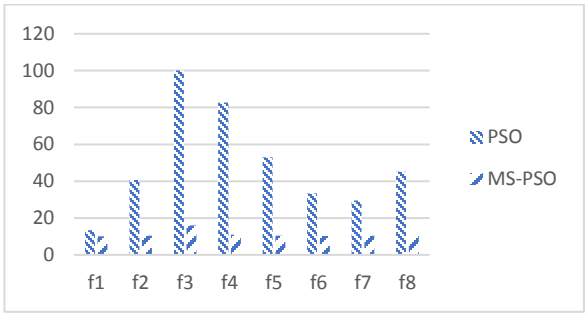

Figure 4. Speedup measure (Ψ) of the compared algorithms with functions scaled to 100 variables. Higher histograms are of better quality. The speed-up of BD-PSO for the function F3 is shortened for better visualization results.

TABLE II. Speedup Ψ measure of the compared algorithms with functions scaled to 100 variables. Dark grey is the algorithm having the highest speedup measure, the last column shows how fast is the proposed BD-PSO compared to the MS-PSO.

Function	MS-PSO	BD-PSO	$\Psi_{DAPDPSO} / \Psi_{MS\text{-}PSO}$
f1	1.3	5.2	3.9
f2	3.9	8.3	2.2
f3	5.3	9.2	1.7
f4	7.5	21.3	2.8
f5	5.1	15.4	3.0
f6	3.2	11.2	3.5
f7	2.8	10.2	3.6
f8	4.4	15.3	3.5
Avg.	4.2	12.0	3.0

With regards to the accuracy (α), as shown in Table 3, BD-PSO is the most accurate method for three out of eight studied problems, and second most accurate for four problems. There is no clear pattern to explain why some problems are more difficult to solve. As an example, the function f3 which is the most expensive and the most difficult to solve is unimodal and convex whereas f8 is the second easiest to solve is multi-modal, irregular and non-separable. Overall, BD-PSO achieved comparable results with MS-PSO for all the problems with the exception of f6 and f7 where MS-PSO was more accurate than BD-PSO; note that f6 is highly multimodal and f7 is unimodal. As it will be

shown in the next sub-section, the performance of BD-BSO tend to improve with a larger number of variables.

TABLE III. ACCURACY (A) OF THE COMPARED OPTIMIZATION ALGORITHMS WITH FUNCTIONS SET TO 100 VARIABLES. DARK GREY IS THE ALGORITHM HAVING THE BEST ACCURACY MEASURE, THE LAST COLUMN SHOWS HOW MANY TIMES FASTER IS IT.

Function	PSO	MS-PSO	BD-PSO
f1	370.0	63.3	61.5
f2	15128.5	3779.4	4156.1
f3	127294.5	99264.0	64218.7
f4	12652.8	1905.1	2803.8
f5	1212.2	1397.3	1159.5
f6	1112.9	30.4	180.0
f7	0.2	0.0	118.7
f8	20.3	20.3	21.4

B. Results with 500 variables

The convergence speed results of BD-PSO, PSO, and MS-PSO when solving problems of 500 variables are shown in Tables 4 and Figure 5. Overall, the execution time of BD-PSO is the fastest; however, when the objective function is expensive such as f3, the effect of the broadcast communication gets attenuated.

TABLE IV. EXECUTION TIME IN SECONDS WITH FUNCTIONS SCALED TO 500 VARIABLES. DARK GREY IS THE FASTEST ALGORITHM; LIGHT GREY IS THE SECOND FASTEST ALGORITHM.

Function	PSO	MS-PSO	BD-PSO
f1	203.7	157	9.3
f2	841.4	128.7	25.4
f3	43493.5	626	9.3
f4	1767.7	144.6	42.7
f5	1170.9	134.9	30.4
f6	725.9	121.9	23.4
f7	616.4	117.9	19.4
f8	203.7	157	24.2

As shown in Table 5 and Figure 6, BD-PSO is in average 6 times faster than MS-PSO with the exception of f3 which is the most expensive function (10 times more expensive than other functions). BD-PSO becomes faster for larger dimensional problems; the speedup difference between BD-PSO and MS-PSO is more accentuated because MS-PSO suffers from a stronger convoy effect at the master particle level where every input vector requires more processing time by the master particle due to a larger number of variables of every solution.

TABLE V. SPEEDUP Ψ MEASURE OF THE COMPARED ALGORITHMS WITH FUNCTIONS SCALED TO 500 VARIABLES. DARK GREY IS THE ALGORITHM HAVING THE HIGHEST SPEEDUP MEASURE, THE LAST COLUMN SHOWS HOW FAST IS THE PROPOSED BD-PSO COMPARED TO THE MS-PSO.

Function	MS-PSO	BD-PSO	$\Psi_{BD\text{-}PSO}$ / $\Psi_{MS\text{-}PSO}$
f1	1.3	21.9	16.9
f2	6.5	33.2	5.1
f3	24.3	41.7	1.7
f4	12.2	41.4	3.4
f5	8.7	38.5	4.4
cf6	6.0	31.0	5.2
f7	5.2	31.7	6.1
f8	8.0	39.7	5.0
Avg.	9.0	34.9	6.0

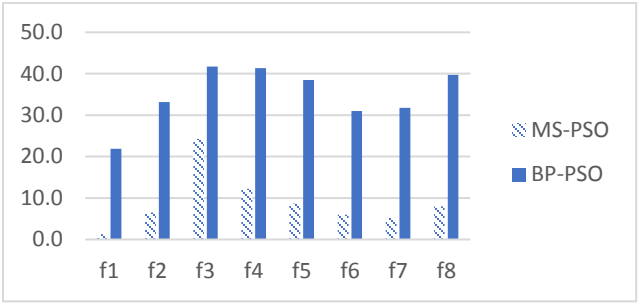

Figure 5. Speedup measure (Ψ) of the compared algorithms with functions scaled to 500 variables. Higher histograms are of better quality.

When solving problems with 500 variables, as shown in Table 6, the accuracy of BD-PSO is improved becoming the best for six out of eight problems. The differences in the result are significant now, so we can say that BD-PSO is significantly more accurate than MS-PSO for four problems, namely, f1, f2, f4, and f6 whereas it achieved a lower accuracy than MS-PSO for the same problems with D =100. In fact, BD-PSO performs poorly only with f7. In brief, based on these results, it can be said that the broadcast communication model led to good results in terms of both accuracy and convergence speed especially for large dimensions D=500. It should be further studied for larger dimension problems to determine whether it is the asynchronous communication model or tye enhanced pseudo-randomness of the distributed model that are enhancing the accuracy of the proposed BD-PSO.

TABLE VI. ACCURACY (A) OF THE COMPARED OPTIMIZATION ALGORITHMS WITH FUNCTIONS SET TO 500 VARIABLES. DARK GREY IS THE ALGORITHM HAVING THE BEST ACCURACY MEASURE, THE LAST COLUMN SHOWS HOW MANY TIMES FASTER IS IT

Function	PSO	MS-PSO	BD-PSO
f1	4,263	1,567	297
f2	982,453	369,373	108,875
f3	2,661,250	99,264	64,219
f4	207,721	68,141	17,740
f5	8,501	8,601	7,753
f6	14,645	5,434	1,759
f7	724,202	4.5	7.3E+22
f8	21.0	21.0	21.5

VI. CONCLUSIONS

In this paper, we have proposed a broadcast distributed particle swarm optimization (BD-PSO) to solve large dimensional optimization problems using MPI. The BD-PSO was in average three times faster than MS-PSO when solving problems D=100 and achieved comparable accuracy results to MS-PSO. However, for D=500, BD-PSO become in average 6 times faster and significantly more accurate than MS-PSO and PSO. These promising preliminary results should lead the community to further study the broadcast communication model in solving large scale and big optimization problems and particularly how can we improve the design of distributed metaheuristics to solve big optimization problems.

HPC cannot only accelerate the convergence speed of metaheuristics but also lead to more accurate results by proposing appropriate algorithms design which should be further studied by the research community. In the, future we aim to study in big optimization problems having expensive objective functions.

REFERENCES

[1] J. Prateek and P. Kar. "Non-convex optimization for machine learning." Foundations and Trends® in Machine Learning vol. 10, no. 3-4, pp. 142-336, 2017.

[2] P. M. Pardalos et al. "A Brief Review of Non-convex Single-Objective Optimization." Non-Convex Multi-Objective Optimization. Springer, pp. 33-42. 2017.

[3] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," Inf. Sci. (Ny)., vol. 237, pp. 82–117, 2013.

[4] G. Gutmann, "Parallel Interaction Detection Algorithms for a Particle-based Live Controlled Real-time Microtubule Gliding Simulation System Accelerated by GPGPU", New Generation Computing, vol. 35, no. 2, pp. 157-180, 2017.

[5] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An Improved Self-Adaptive Differential Evolution Algorithm for Optimization Problems," IEEE Trans. Ind. Informatics, vol. 9, no. 2, pp. 89–99, 2012.

[6] S.J Nanda, and P. Ganapati. "A survey on nature inspired metaheuristic algorithms for partitional clustering", Swarm and Evolutionary computation, vol. 16, pp. 1-18, 2014.

[7] E. Talbi, Metaheuristics from Design to Implematation. John Wiley & Sons Publication Inc, 2009.

[8] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: recent advances and new trends," Int. Trans. Oper. Res., vol. 20, no. 1, pp. 1–48, 2013.

[9] Xiaodong, L., Tang, K., Mohammad, N. O., Zhenyu, Y., & Kai, Q. Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Gene, vol. 7, no. 33, 1–8, 2013.

[10] K. Yaji et al. "Large-scale topology optimization incorporating local-in-time adjoint-based method for unsteady thermal-fluid problem." Structural and Multidisciplinary Optimization, pp. 1-6, 2018.

[11] R. C. Eberhart and J. Kennedy, "Particle Swarm Optimization," in 1995 IEEE International Conference on Neural Networks, pp. 1942–1948, 1995.

[12] Y. Zhang, S. Wang, G. Ji, Y. Zhang, S. Wang, and G. Ji, "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications," Math. Probl. Eng., vol. 2015, pp. 1–38, 2015.

[13] X. Li, and Y. Xin. "Cooperatively coevolving particle swarms for large scale optimization", IEEE Transactions on Evolutionary Computation, vol. 16, no. 2, pp. 210-224, 2012.

[14] R. Impagliazzo, R. Shaltiel, and A. Wigderson, "Near-optimal conversion of hardness into pseudo-randomness," in In IEEE 40th Annual Symposium on Foundations of Computer Science, pp. 181–190, 1999.

[15] Gong, Yue-Jiao, et al. "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art." Applied Soft Computing, vol. 34, pp. 286-300, 2015.

[16] A. AitZai and M. Boudhar, "Parallel branch-and-bound and parallel PSO algorithms for job shop scheduling problem with blocking," Int. J. Oper. Res., vol. 16, no. 1, pp. 14–37, 2013.

[17] J. Kaur, S. Singh, and S. Singh, "Parallel Implementation of PSO Algorithm using GPGPU," in The Second International Conference on Computational Intelligence & Communication Technology, Hyderabad, India pp. 155–160, 2016.

[18] J. Shiyuan, D. Damian, and Q. Zhihua, "Parallel Particle Swarm Optimization (PPSO) on the Coverage Problem in Games, Pursuit-evasion," in HPC '12, Symposium on High Performance Computing, Orlando, Florida, 2012.

[19] E. Ahmed, K. R. Mahmoud, S. Hamad, and Z. T. Fayed, "Real Time Parallel PSO and CFO for Adaptive Beam-forming Applications," in PIERS Proceedings, 2013, pp. 816–820.

[20] Z. Fu, "Research on the Prediction of the E-commerce Profit Based on the Improved Parallel PSO-LSSVM Algorithm in Cloud Computing Environment," vol. 9, no. 6, pp. 369–380, 2016.

[21] I. Aljarah and S. A. Ludwig, "Towards a Scalable Intrusion Detection System based on Parallel PSO Clustering Using MapReduce Categories and Subject Descriptors," in Gecco'13, 2013.

[22] D. Liu, Y. Liu, Y. Liu, and Z. Xiang, "A Parallelized Multi-objective Particle Swarm Optimization model to design soil sampling network," in Geoinformatics (GEOINFORMATICS), 2012 20th International Conference on, pp. 1–6, 2012.

[23] T. Gonsalves and A. Egashira, "Parallel Swarms Oriented Particle Swarm Optimization", Appl. Comp. Intell. Soft Comput., vol. 2013, no. 756719, pp. 1–8, 2013.

[24] G. W. Zhang et al., "Parallel Particle Swarm Optimization Using Message Passing Interface," in Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, 2015, vol. 1, no. 61125205, pp. 55–64.

[25] L. Mussi, Y. S. G. Nashed, and S. Cagnoni, "GPU-based asynchronous particle swarm optimization," in Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11, pp. 1555–1562, 2011.

[26] A. Ayari & S. Bouamama "A new multiple robot path planning algorithm: dynamic distributed particle swarm optimization", Robot. Biomim, vol. 4, no. 8, 2017.

[27] K. Ishikawa et al., "Consensus-Based Distributed Particle Swarm Optimization with Event-Triggered Communication", IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E101-A, No.2, pp.338-344, 2018

[28] J. M. Hereford, "A particle swarm algorithm for multiobjective design optimization," in 2006 IEEE Congress on Evolutionary Computatio, 2006, vol. 1, pp. 1678–1685.

[29] Ş. Gülcü and H. Kodaz, "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," Eng. Appl. Artif. Intell., vol. 45, pp. 33–45, 2015.

[30] S. N. Omkar, A. Venkatesh, and M. Mudigere, "MPI-based parallel synchronous vector evaluated particle swarm optimization for multi-objective design optimization of composite structures," Eng. Appl. Artif. Intell., vol. 25, no. 8, pp. 1611–1627, 2012.

[31] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," Inf. Sci. (Ny)., vol. 178, no. 15, pp. 3096–3109, 2008.

[32] S. Mostaghim, "Parallel multi-objective optimization using self-organized heterogeneous resources," in Parallel and Distributed Studies in Computational Intelligence, Springer., vol. 269, Berlin Heidelberg, 2010, pp. 165–179.

[33] K. E. Parsopoulos, "Parallel cooperative micro-particle swarm optimization: A master-slave model," Appl. Soft Comput. J., vol. 12, no. 11, pp. 3552–3579, 2012.

[34] S. Chen, J. Montgomery, and A. Bolufé-Röhler, "Some measurements on the effects of the curse of dimensionality," in Proceedings of the 2014

conference companion on Genetic and evolutionary computation companion - GECCO Comp '14, 2014, no. July, pp. 1447–1448.

[35] M. Pant, R. Thangara, and A. Ajith, "Particle swarm optimization: performance tuning and empirical analysis.","" Found. Comput. Intell., vol. 3, pp. 101–128, 2009.

[36] A. McNabb and K. Seppi, "Serial PSO results are irrelevant in a multi-core parallel world," in Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, 2014, pp. 3143–3150.

[37] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in Swarm Intelligence Symposium, 2007, pp. 120–127.

[38] A. H. Karp and H. P. Flatt., "Measuring Parallel Processor Performance," Commun. ACM, vol. 33, no. 5, pp. 539–543, 1990.

AUTHORS PROFILE

Dr. Farid Bourennani is an Assistant Professor in the Faculty of Computing and Information Technology at the University of Jeddah. His research projects focus on metaheuristics optimization methods to solve real engineering problems. Furthermore, Dr. Bourennani is actively working in data science especially in heterogeneous data mining area that he pioneers. He has published over 20 peer reviewed articles in international journals and conferences. During his PhD, Farid Bourennani was twice the recipient the Jeffrey Boyce Engineering Award which is the highest and most prestigious award at University of Ontario Institute of Technology, Canada. He was nominated for PhD Alumni Gold Medal with a GPA of 4.3/4.3, and he received other awards such Ontario Graduate Scholarship (OGS), and Best Paper Award at the DBKDA'09 conference among others. Prior to that, Dr. Bourennani worked several years in the industry in north America with known companies such as IBM, Bell Canada, Sabic, Einstein, Alis Technology, Look Communication, and AxessUP among others.