ISSN 2278-3091
Volume 8, No.2, March - April 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse03822019.pdf
https://doi.org/10.30534/ijatcse/2019/03822019

A modified Particle Swarm Optimization algorithm linking dynamic neighborhood topology to parallel computation

Maria Zemzami[1], Norelislam Elhami[2], Mhamed Itmi[3], Nabil Hmina[4]
[1]LITIS-INSA-Rouen, France, maria.zemzami@gmail.com
[2]LGS-ENSA-Kenitra, Morocco, norelislam@outlook.com
[3] LITIS-INSA-Rouen, France, itmi@insa-rouen.fr
[4]LGS-ENSA-Kenitra, Morocco, hmina5864@gmail.com

ABSTRACT

In this paper, a novel approach is considered, based on Particle Swarm Optimization (PSO) technique, using two concepts: evolutionary neighborhood topology associated to parallel computation for complex optimization problems. The idea behind using dynamic neighborhood topology is to overcome premature convergence of PSO algorithm, by well exploring and exploiting the search space for a better solution quality. Parallel computation is used to accelerate calculations especially for complex optimization problems. The simulation results demonstrate good performance of the proposed algorithm in solving a series of significant benchmark test functions.

Key words: Optimization, metaheuristic, PSO, Dynamic neighborhood, Parallel computing.

1.INTRODUCTION

After the ongoing evolution of material resources in IT, computers had their number of processors / cores increased in recent years to compensate for limits of the increasing power for a single processor and obtain an acceleration factor, since with more computing power a problem could be solved quickly. To fully exploit this computing power, it should implement applications capable of performing several tasks in parallel [1].

Threads are the technology used in Java to make multitasking applications. We were interested in this technology to take advantage of parallelism in terms of reduction in computing time and good use of material resources of the machine.

PSO is a metaheuristic designed to finding the optimum of a function at a reasonable time, except for large instances where scientific computing is intensive requiring a considerable computing time. The use of appropriate parallel models reduces the computation time and gives better results than the sequential models [2] [3]. Escaping the premature convergence of the method is also a key point on which several researchers conducted their studies and suggested several versions [4-7]. The model we suggest in this paper is a version based on the PSO algorithm using threads for parallel computing, and a new concept of dynamic neighborhoods to avoid premature convergence of the method.

In our experimentations, the tests conducted on the program have given satisfactory results of our model compared to the basic PSO algorithm.

The remainder of this paper is organized as follows: Section 2 contains the description of PSO method. Section 3 is a presentation our proposed approach. The testing and interpretation of results will be subject to Section 4, followed by a conclusion.

2.OVERVIEW OF PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization is a very known metaheuristic, proposed in 1995 by the two American inventers James Kennedy “psychologist” and Russel Eberhart “electrical engineer” in order to solve discrete and continuous optimization problems [8].

It is inspired from the social behavior of individuals evolving in swarm, i.e. the "social interactions" between "agents" called "particles" representing a "swarm", in order to achieve a given goal in a common search space where each particle has a certain capacity for memorizing and processing information.

Unlike other evolutionary algorithms such as the genetic algorithm where the search for the optimal solution evolves by competition between individuals using operators of crosses and mutations, the PSO algorithm uses cooperation between individuals (which makes the method very powerful).

2.1 PSO algortithm

PSO algorithm is a stochastic process, where the particles move around a search space in search of the optimum. It is proposed by [8], starts with a random initialization of the particles in their search space, by attributing their initial positions and velocities. At each iteration of the algorithm particles move and the objective functions (fitness) of particles are calculated in order to calculate the global best position Gb. The update of Pb and Gb is made at each iteration according to the algorithm cited in Figure 1. The process is repeated until the stopping criterion is met.

Algorithm 1 : Basic PSO Algorithm

```
begin
Initialization()
Initialize the swarm giving initial locations and velocities
stochastically assigned
while stopping creterion is not satisfied do
    Fitness()
    Evaluate the particle scores using a Fitness function
    Updating()
    According to Fitness values Update the Pb and Gb
    Evolution()
    Move particle according to the evolution equations
    Estimation()
    The output is the Gbest
end
end
```

Figure 1: Basic PSO algorithm

2.2 Configuration of the method

There are several parameters involving and influencing the PSO performance. The choice of these parameters remains critical and generally depends on the optimization problem [9] [10] but has a high influence on the convergence of the algorithm. Among these parameters:
- The dimension of the problem;
- Acceleration coefficients;
- The inertia weight;
- The constriction factor;
- The concept of neighbourhoods;
- The number of particles;
- The disposition of particles;
- The stopping criterion;
- The maximum speed;

We will then focus on the last four parameters.

A. The number of particles

One of the key PSO parameters is the number of particles, it greatly influences the performance of the algorithm, especially in terms of computational time, since the presence of each particle in the algorithm causes a calculation: evaluation of the position and the movement of the particle.
The number of particles allocated to solving a problem depends on several parameters, namely: the dimension of the problem to be optimized (the size of the search space), the ratio between the computing capacity of the machine and the time maximum research, and particularly the complexity of the optimization problem.
The choice of an adequate value for this parameter is not an easy task, since there is no rule to determine it, only a massive experimentation by doing many tests makes it possible to acquire the necessary experience to the apprehension of this parameter.

B. The disposition of particles

Before starting the algorithm, the positions of the particles and their initial velocities must be initialized randomly according to a uniform law on [0..1]; this initial disposition affects the next movement of each particle and thus the convergence of the algorithm, especially in the case where we have geographical neighborhoods.
However, there is a set of automatic position generators, to assign different positions to the entire swarm.
The SOBOL sequence generator is one of the most efficient in this field, for a homogeneous disposition of particles in a n-dimensional space [11].

C. The stopping criterion

The stopping criterion is an important parameter for any optimization method. It differs depending on the optimization problem and the constraints of the user, it is strongly recommended to provide the algorithm with this parameter since the convergence to the optimal solution is not guaranteed in all cases even if the experiments denote the great performance of the method. As a result, several studies have been conducted in this direction [12], different propositions have taken place: the algorithm must then execute as long as one of the convergence criteria has not been reached. This can be: the maximum number of iterations; the global optimum is known a priori; we can define an "acceptable accuracy".
Other stopping criteria can be used depending on the optimization problem and user constraints.

D. The maximum speed

The maximum speed was proposed by [13] in 1996, as a solution to the problem of deflecting particles during their movement.
The objective was to limit the particle velocity by the interval [-vmax, vmax] in order to control the movement of each particle in the search space.
The introduction of Vmax allowed better control of particle motion for a more optimal convergence.
The use of this parameter has resulted in several publications [14]-[18].

3.THE PROPOSED APPROACH

This section presents a new model based on PSO algorithm, using a novel dynamic neighborhood topology associated to parallel computation for complex optimization problems.
The idea behind the combination of these two concepts came after a deep study of the PSO algorithm and its different versions (improvements).
The use of a static neighborhood is less expensive in terms of computation time (there is no updating of the neighborhoods at each iteration); the neighborhoods remain the same from the beginning of the program until its end. As well as their easy implementation, but for our approach, we opted for a version using a dynamic neighbourhood.
The use of a dynamic neighborhood, allows a better exploration and exploitation of the search space in order to improve the solution quality, but it is expensive in terms of computation time, since it is necessary to update the neighborhoods (at each iteration of the algorithm).
To overcome this constraint, we parallelize the calculations; this parallelization has improved the proposed approach in terms of computation time.

In the literature, several authors have proposed parallel models of the PSO method [19]-[21], the one we implemented in our approach allows the parallelization of calculations using the concept of threads in Java: each thread deals with the PSO processing for its neighborhood.
Below the flowchart of the proposed approach Figure 2.
For other models based on PSO algorithm the reader is referred to [22]-[24].

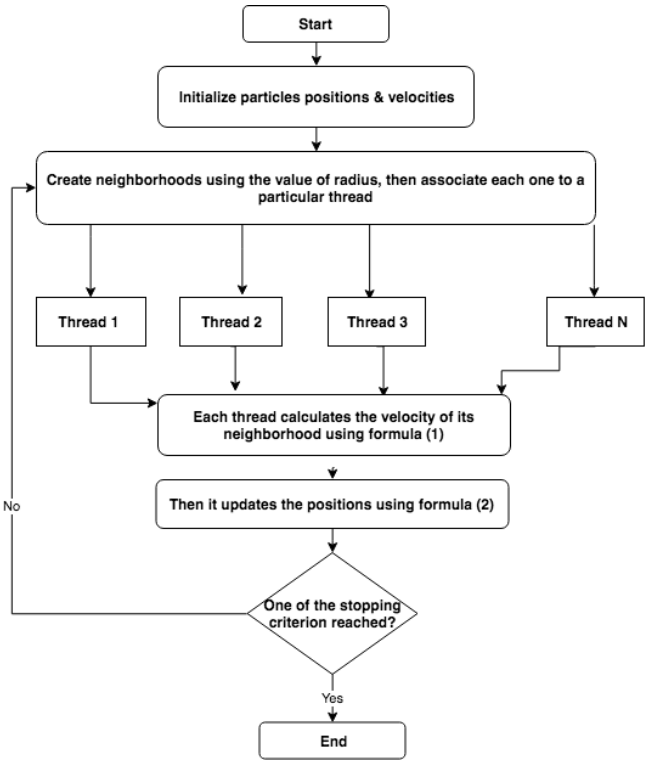

Figure 2: PPSO model flowchart

3.1 Used settings

Each parameter of the PSO algorithm has a major influence on the behavior of particles and thus the convergence of the algorithm; and even if the PSO method provides satisfactory results, choosing the right parameter of the method remains a critical issue as one of the keys to success for any PSO algorithm.
In the previous section, we presented some parameters that influence the behavior of particles in their travels in search of the optimum. The parameter set that we have developed in our model consists of the use of multiple variable parameters that can be modified from the user interface dedicated to this; everything depends on the requirements of the optimization problem.
We have implemented a version with the inertia factor; the value of the latter is configurable from the user interface. Another version with the constriction coefficient, which is computed automatically.
C1 and C2 acceleration coefficients are also variables, their default values: C1 = 1.25 and C2 = 2.25, C3=1.25, providing well results in the majority of experiments, but that can be changed from the user interface. For communication topologies used, the three topologies "star, radial, and ring" are implemented, and may be selected from the user interface. The principle of creating neighborhoods, parallel processing, the stopping criteria and the algorithm will be detailed below.

3.2 Evolutionary Neighbourhoods

The neighborhoods are dynamic spheres, with each iteration the number of particles in different spheres changes according to the new positions of the particles and the radius value.
The creation of the spheres is as follows: We initialize the particle positions, we specify the initial value of the radius, and it is considered a first particle Pc. It then represents the center of the sphere S of radius r. Pa is a particle adjacent to particle Pc if Pa Euclidean distance to Pc is less than or equal to the value of the radius r. Otherwise, it becomes the center of a new sphere. Every new particle has its belonging which is studied with respect to the different spheres created before agreeing to create a new sphere. Furthermore, if the number of field is reduced (preset number) then the common radius of the spheres is reduced significantly. The peculiarity of the neighborhoods of our model is that we benefit from the advantages of the concept of neighborhood in the sharing of information and cooperation between the sub-swarms, without falling into the trap of premature convergence. In the model of the PSO algorithm with neighbors, sharing Pn "the best of each neighborhood" is done at each iteration; and based on a comparison of all the Pn obtained, the best of all the Pg swarm is defined. That said, if a particle of a neighborhood links to a web- site developer (containing a good solution), and it turns out to be best in it neighborhood at the end of the iteration the information will be propagated, and this particle will be declared the best of the whole swarm, so it will influence the displacement formula of all particles, which will lead to this site. We assume that this site contains a local optimum, and that there is obviously the optimal solution somewhere in the overall search space, but taking into consideration the influence of the information propagated to each iteration in the displacement of particles, the latter link to the wrong path, which leads to premature convergence. What we are proposing in our model is that the various neighborhoods look independently for the solution of the of the Gb value. Each particle moves according to its best Pb value, and the best in it Pn neighborhood. Our model always respects the basic principle of the PSO algorithm based on cooperation between the particles, and the sharing of information that still exist, since the neighborhoods are dynamic. In every iteration particles change their neighborhoods and thus they broadcast their information in new neighborhoods.
Failure sharing Pg (overall best known position) with each iteration enables better use of space research and gives more opportunity for particles to avoid the anomaly of algorithm's premature convergence.

3.3 Parallel computation

Our parallel approach based on PSO algorithm, consists of launching a set of processes (threads) simultaneously. Each thread is responsible for processing a set of particles for all iterations until stopping criterion is reached. At the end of each iteration; a thread synchronization is done to assess the results of each neighbourhood and update the neighbourhoods to begin a new iteration.

3.4 Stopping criteria

To minimize computational time and obtain satisfactory results, we opted in our program for three stopping criteria:

1) If the maximum number of iterations without improvement is reached a specific number,
2) Or, when the fixed radius reached a precision,
3) Or, when the fixed distance of Gbest reached a precision,

Regarding the first criterion, we specify a number of iterations after which there is no remarkable improvement in the solution and we stop the program. The second criterion relates to a value specifying the minimum radius allowed, if this value is reached the program execution stops. The third criterion concerns the value of the best position of the whole swarm, if the distance between the value of the best position at iteration t and the best position at iteration t + 1 is equal to a specified accuracy, that is to say that there is no significant improvement in the solution then we stop the program execution. All these criteria are variables, configurable from the user interface, depending on the problem to be optimized.

3.5 Algorithm framework

The main steps of our algorithm are as follows:

Step 1: Generate randomly a set of particles and their positions and speeds.

Step 2: Creating neighborhoods on the basis of the radius value.

Step 3: The processing of each neighborhood is attributed to one of the created thread.

Step 4: Each thread evaluates the velocity and the position of all its own particles.

Step 5: Update the neighborhoods according to the new particles' positions and radius' value.

For X number of iterations do
While (stopping criterion not reached)
If the number of neighborhoods is less than Z
Divide the radius by 2
End if
Create spherical neighborhoods based on the radius value
For every N neighborhoods a Thread do
For each particle of a neighborhood
If new Localbest is better than old Localbest
Update Localbest of the neighborhood
End if
End for
End for
End While
End for

Step 6: If the stopping criterion is satisfied, stop, otherwise go to step 2.

3.6 Pseudo code

4.DESCRIPTION OF OUR EXPERIMENT AND RESULTS

The modification in the basic PSO algorithm for our approach consists of three categories: a new version of dynamic neighborhood, parallel computation, and adjustment of the PSO parameters. These modifications of PPSO algorithm enhance its performance.

4.1 Benchmark problems

To test the optimality of our proposed approach PPSO, we used a set of test functions; they are created specifically to test the performance of different optimization methods.
For this paper, we have chosen to present the results of 10 test functions (see Table 1), these so-called complex functions (contain a large number of local optimum) and high dimensions.

Table 1: Description of the used functions in our experiments

Funtion	Range	f_{min}	Dim
f_1 Sphere	±5.12	0	30
f_2 Griewank	±600	0	30
f_3 Rosenbrock	±30	0	30
f_4 Rastring	±5.12	0	30
f_5 Schwefel	±500	0	30
f_6 Ackley	±32	0	30
f_7 Michalewicz	$\pm\pi$	-9.66015	10
f_8 Shubert	±10	-186.739	10
f_9 Step	±100	0	30
f_{10} Himmelblau	±30	-3.78396	2

4.2 Experimental Settings

As for all metaheuristics, PSO has a set of parameters that must be defined by the user at the beginning of the program, i.e. the number of particles, the size of the problem to be optimized, the initial positions and velocities of the particles, the communication topology, the values of acceleration coefficients, the number of iterations... and of course for our model, other parameters are added: the value of the radius, the values relating to the stopping criteria, Etc.
For this study, which consists of an experiment with a set of medium-sized problems: 2, 10 and 30 dimensions, the list of the used PSO parameters which give satisfactory results are taken from the study [25].
For our PPSO model, the parameters are defined in

accordance with our approach, for example: The inertia factor is variable and smaller for greater local search capacity. As well as the communication topology, ring is the best topology to implement in our approach for a better exploration.

For the experimentation conducted to PPSO program, each thread is assigned a neighborhood processing (the number of used threads is equal to the number of neighborhoods created). So, the number of used threads depends on the objective function, i.e. the dimension of the search space. The same thing for the stopping criteria, the value of each criterion is chosen depending of the optimization problem.

The choice of radius value is very important, because neighborhoods are created using this value; (a very large radius value is equal to a small number of neighborhoods, while a small value is equal to many. So the choice of this criterion remains critical and depends on the problem to be optimized).

For each test, the results for 1000 runs of each objective function were averaged. To demonstrate the quality of our Java code, we used JUnit framework for the implementation and execution of automated unit tests. Throughout the unit test development process were made on the different classes / components of the program to ensure that the code still meets the needs even after any changes.

4.3 Results

The carried out experiments are based on the launching of the PSO parallel processing on a set of particles being positioned in dynamic neighborhoods in search of the "minimum" optimum of the objective function. The graphs below show the detail of the average of results namely the values of the execution time in seconds, the SR (Success Rate): the success rate is the percentage of function convergence to the right solution, and SD (Standard Deviation) represents the standard deviation $SD = \sqrt{1/n(\sum_{i=1}^{n}(Xi - X^*)^2)}$ where X*: the optimal solution and Xi: the solution found for each test and for the sequential and parallel program PSO model on a set of ten functions.

According to the results, we can say that the PPSO provides the optimal solution with a higher probability and the computation time in PPSO is lower than the sequential PSO. The graphical results are illustrated in the figures below (Figure 3, Figure 4 and Figure 5).

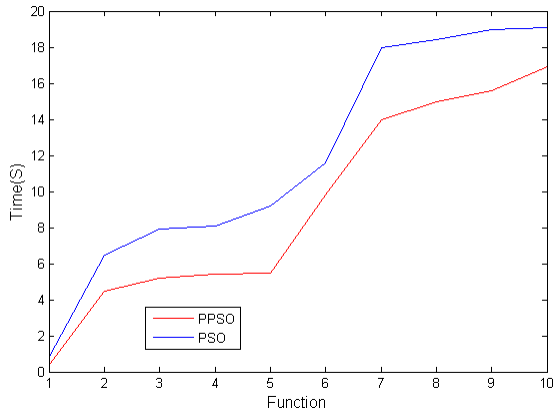

Figure 3: Performance curves of the computation time for PSO and PPSO

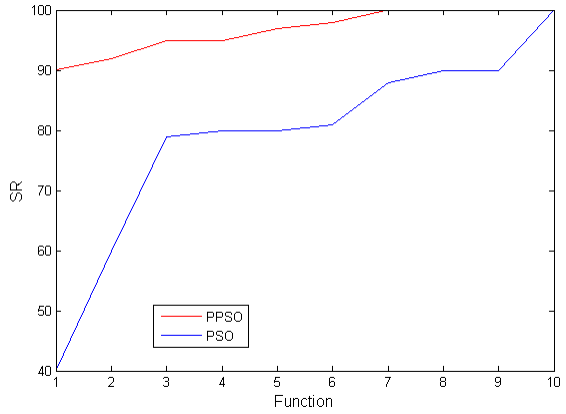

Figure 4: Performance curves of the Success Rate for PSO and PPSO

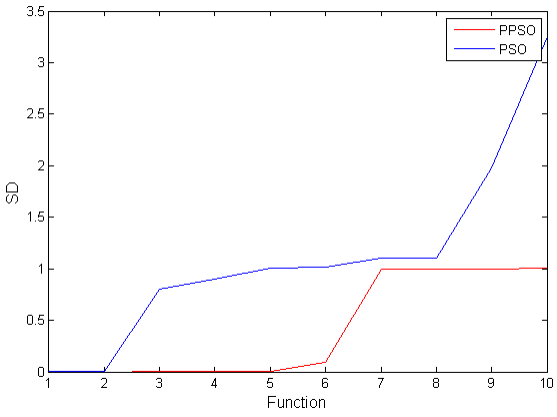

Figure 5: Performance curves of the Standard Deviation for PSO and PPSO

5.CONCLUSIONS AND FUTURE WORK

This paper contains a description of the implementation of a parallel approach with evolutionary neighborhoods based on the PSO algorithm. PSO is a stochastic process where the particles move around a search space in search of the optimum. Although the method is well known for its

robustness in solving very complex optimization problems, the latter has two major weaknesses: premature convergence and high running time.

Several models based on the PSO algorithm have been proposed to improve the method, and to avoid these two defects, either by adding new parameters, by hybridizing with other methods or by introducing the parallelization.

For our PPSO approach coupling two concepts: evolutionary neighborhood and parallel computation. The obtained results from the experimentation of 10 test functions prove the effectiveness of the PPSO and show remarkable efficiency in terms of reduced time and optimality convergence.

Finally, in the future we intend to test the PPSO on high dimension functions, to study other variants of the proposed parallel model and for various real optimization problems.

ACKNOWLEDGEMENT

This research is supported by « XTERM »: Complex Systems, Territories Intelligence and Mobility, co-financed by the European Union with the European regional development fund (ERDF) and Normandy Region.

REFERENCES

1. P.Sowndarya Mala, N.M.Ramalingeswara Rao, V.Sreevani, M.Sai. **Analysis and Reduction of high power consumption using parallel prefix adder.** International Journal of Advanced Trends in Computer Science and Engineering, vol.7, no.6, pp. 163- 165. 2018. https://doi.org/10.30534/ijatcse/2018/21762018
2. Y.Zhou and Y.Tan. **GPU-Based Parallel Particle Swarm Optimization.** In: Proceedings of the *IEEE Congress on Evolutionary Computation*, vol. (9), pp. 1493–1500, 2009. https://doi.org/10.1109/CEC.2009.4983119
3. M.Waintraub, R. Schirru and C. Pereira. **Parallel Particle Swarm Optimization Algoritms in Nuclear Problems.** In: International Nuclear Atlantic Conference – INAC, 2009.
4. Ting, T.-O., Rao, M.V.C., Loo, C.K., Ngu, S.-S. **A New Class of Operators to Accelerate Particle Swarm Optimization.** In: Proceedings of *the IEEE Congress on Evolutionary Computation*, vol. (4), pp. 2406– 2410, 2003.
5. Liu, H., Abraham, A., Zhang, W. **A Fuzzy Adaptive Turbulent Particle Swarm Optimization.** International Journal of Innovative Computing and Applications 1(1), 39–47, 2007. https://doi.org/10.1504/IJICA.2007.013400
6. Paquet, U., Engelbrecht, A.P. **A New Particle Swarm Optimizer for LinearlyConstrained Optimization.** In: Proceedings of the *IEEE Congress on Evolutionary Computation*, vol. (1), pp. 227–233, 2003.
7. Parsopoulos, K.E., Plagianakos, V.P., Magoulus, G.D., Vrahatis, M.N. **Objective Function "Strectching" to Alleviate Convergence to Local Minima.** Nonlinear Analysis, Theory, Methods and Applications 47(5), 3419–3424, 2001. https://doi.org/10.1016/S0362-546X(01)00457-6
8. J. Kennedy and R. Eberhart. **Particle Swarm Optimization**. In: Proceedings of the *IEEE International Joint Conference on Neural Networks*, IEEE Press, vol. 8, no. 3, pp. 1943–1948, 1995.
9. K. E. Parsopoulos and M. N. Vrahatis. **Recent approaches to global optimization problems through particle swarm optimization.** In: Natural Computing: an international journal, 1(2-3), pp.235-306, 2002.
10. M.E. Hyass and P. Hyass. **Good Parameters for Particle Swarm Optimization.** In: Laboratories Technical Report no. HL1001. 2010.
11. P. Bratley and B. L. Fox**. Algorithm 659: Implementing Sobol quasirandom sequence generator**. ACM Trans. Math. Software 14, p.88-100, 1988. https://doi.org/10.1145/42288.214372
12. K. Zielinski and R. Laur. **Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimization.** In: Advanced in Differential Evolution, the series Studies in Computational Intelligence Vol. 143, pp. 111-138 Springer, Berlin Heidelberg. 2008. https://doi.org/10.1007/978-3-540-68830-3_4
13. R.C. Eberhart, P. Simpson, R. Dobbins**. Computational PC Tools.** Chapter 6, AP Professional. pp. 212-226, 1996.
14. R.C. Eberhart, Y. Shi. **Comparing inertia weights and constriction factors in particle swarm optimization.** Proceedings of the *6th IEEE Congress on Evolutionary Computation*, IEEE Press. pp. 84-88, 2000.
15. H.Y. Fan, Y. Shi. **Study on Vmax of particle swarm optimization.** Proceedings of the *2001 Workshop on Particle Swarm Optimization*, Indiana University-Purdue University Indianapolis Press. 2001.
16. K. Deep, J. C. Bansal. **Hybridization of particle swarm optimization with quadratic approximation.** OPSEARCH. Vol. 46, N° 1, pp. 3-24, 2009. https://doi.org/10.1007/s12597-009-0002-5
17. X. Cai, Y.Tan. **A study on the effect of vmax in particle swarm optimization with high dimension.** International Journal of Bio-Inspired Computation (IJBIC). Vol. 1, N°. 3, pp. 210 - 216, 2009. https://doi.org/10.1504/IJBIC.2009.023816
18. J. Barrera, C. A.C. Coello. **Limiting the velocity in particle swarm optimization using a geometric series.** Genetic And Evolutionary Computation Conference, Proceedings of the *11th Annual conference on Genetic and evolutionary computation*, pp. 1739-1740, 2009. https://doi.org/10.1145/1569901.1570135
19. P. Rabanal, I. Rodríguez and F. Rubio. **Parallelizing Particle Swarm Optimization in a Functional Programming Environment.** In Algorithms2014 : vol. 7, pp. 554–581, 2014. https://doi.org/10.3390/a7040554
20. K. Byung-I and G. Alan. **Parallel asynchronous particle swarm optimization.** International Journal For

Numerical Methods In Engineering, vol. 67, pp. 578-595, 2006.
https://doi.org/10.1002/nme.1646

21. J. Chang, S. Chu, J. Roddick and J. Pan. **A Parallel Particle Swarm Optimization Algorithm With Communication Strategies.** In : Journal of Information Science and Engineering, 2005.
22. M. Zemzami, A. Elhami, A. Makhloufi, N. Elhami, M. Itmi, and N. Hmina: **Electrical Power Transmission Optimization based on a New Version of PSO Algorithm**. (Published 22/02/17 DOI: 10.21494/ISTE.OP.2017.0127). 2017.
https://doi.org/10.21494/ISTE.OP.2017.0127
23. M. Zemzami, N.Elhami, M.Itmi and N.Hmina. **A New Parallel Approach For The Exploitation Of The Search Space Based On PSO Algorithm**. In *IEEE 4th International Colloquium in Information Science and Technology (CIST'16).* Tangier. Morocco. Scopus Indexed, 2016.
https://doi.org/10.1109/CIST.2016.7805024
24. M. Zemzami, A. Elhami, A. Makhloufi, N. Elhami, M. Itmi, and N. Hmina. **Applying a new parallelized version of PSO algorithm for electrical power transmission.** International Conference on Materials Engineering and Nanotechnology (ICMEN'17). Kuala Lumpur, Malizia. Indexed by Ei Compendex and Scopus, 2017.
https://doi.org/10.1088/1757-899X/205/1/012032
25. Perdesen, M. E. H. **Good parameters for partcile swarm optimization**. Hvass Lab., Denmark, Tech. Rep. HL1001. 2010.