# An improved model for complex optimization problems based on the PSO algorithm

Maria ZEMZAMI
LGS Lab- Sultan Moulay Slimane University. Beni Mellal- Morocco

maria.zemzami@gmail.com

Norelislam EL HAMI
LGS Lab- ENSA- Ibn Tofail University. Kenitra - Morocco

norelislam@outlook.com

Nabil HMINA
LGS Lab- Sultan Moulay Slimane University. Beni Mellal- Morocco

hmina5864@gmail.com

***Abstract*— recognized for many years as an effective method for solving difficult optimization problems, the Particle Swarm Optimization method however has drawbacks, the most studied are: the high computing time and the premature convergence. This paper highlights a new variant of the PSO method aimed at avoiding these two drawbacks of the method. This variant combines two approaches: the parallelization of the computational method and the organization of appropriate neighborhoods for the particles. The performance evaluation of the proposed model was made based on an experiment on a series of test functions. In the light of the analysis of the obtained results, we observe that the proposed model gives better results than those of the classic PSO in terms of quality of the solution and of the calculation time.**

***Keywords—PSO, Optimization, Parallelization, neighborhood.***

## I. Introduction

Optimization problems in the real world are usually complex. They not only contain the terms of constraints, of simple / multiple objectives, but their modeling is constantly evolving. Their resolution and iterative evaluation of objective functions require a long CPU time. The high computational cost for solving these problems has prompted the development of optimization algorithms with parallelization.

The Particle Swarm Optimization (PSO) algorithm is one of the most popular algorithms based on swarm intelligence, which is enriched with robustness, simplicity and global search capabilities. However, one of the main obstacles of the PSO method is its susceptibility to get trapped in local optima and; Like other evolutionary algorithms, the performance of PSO deteriorates as the size of the problem increases. Therefore, several efforts are made to improve its performance. Different scenarios are developed, either by adding new parameters to the basic algorithm [1], by hybridizing it with other meta-heuristics [2], or by proposing parallelization scenarios [3] - [6].

The basic architecture of PSO inherits a natural parallelism, and the responsiveness of fast processing machines has made this task very convenient. Therefore, parallel models based on the PSO meta-heuristic have become very popular because parallelization offers an excellent path to improve system performance.

The present paper has five sections including the introduction. In the next section, a description the PSO algorithm is given. In section 3 our parallel approach of PSO is discussed. Section 4 describes experiment and results. The chapter finally concludes with Section 5.

## II. Review of Particle Swarm Optimization

### *A. Concept*

Particle swarm optimization (briefed as PSO) is a stochastic optimization meta-heuristic; based on population solutions, proposed in 1995 by James Kennedy (socio-psychologist) and Russel Eberhart (electrical engineer) for the resolution of the optimization problems, particularly continuous variable problem. [7].

PSO is based on the social behavior of individuals evolving in swarms, i.e., the "social interactions" between "agents" called "particles" representing a "swarm", in order to achieve a given objective in a common research space. Where each particle has a certain capacity for memorizing and processing information. In PSO, social behavior is modeled by a mathematical equation to guide particles during their process of movement [7].

The movement of a particle is influenced by three components: the inertia component, the cognitive component and the social component. Each of these components reflects part of the particle's movement equation.

1. The inertia component: the particle tends to follow its current direction of travel;

2. The cognitive component: the particle tends to move towards the best site through which it has already passed;

3. The social component: the particle tends to move towards the best site reached by its neighbors.

### *B. Algorithm*

The basic algorithm of the PSO method proposed by [7] begins with a random initialization of the particles in their search space, by assigning them an initial position and speed. With each iteration of the algorithm, the particles move according to the displacement equations, and the objective functions of the particles (fitness) are calculated so that the best position of all can be calculated.

The best position of each particle and the best position of the swarm are updated with each iteration. The process is repeated until the stop criterion is met (see Fig. 1).

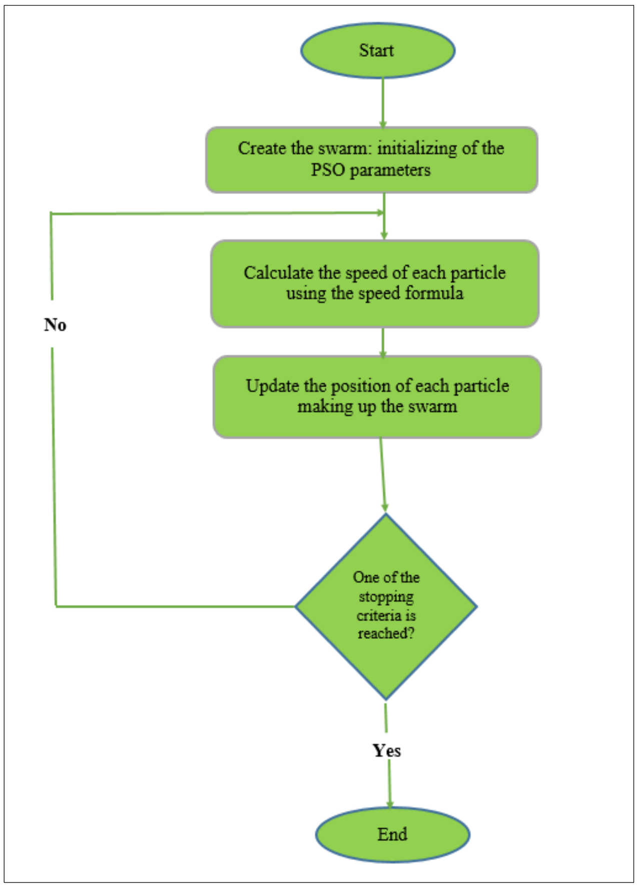
Start
Create the swarm: initializing of the PSO parameters
Calculate the speed of each particle using the speed formula
Update the position of each particle making up the swarm
One of the stopping criteria is reached?
No
Yes
End

Fig. 1. Basic PSO Flowchart

### C. *PSO configuration*

Like any meta-heuristic, PSO has a set of parameters that intervene and influence its performance. The choice of these parameters remains critical and generally depends on the problem posed [8] [9] but has a great influence on the convergence of the algorithm. Among these parameters, we can mention:

- **Number of particles:** One of the key parameters of the PSO method is the number of particles; it greatly influences the performance of the algorithm, especially in terms of calculation time, since the presence of each particle in the algorithm causes a calculation: evaluation of the position and displacement of the particle [10]. The quantity of particles allocated to the resolution of 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 computational capacities of the machine and the time maximum of research and especially 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 experiment by making many tests allows acquiring the necessary experience to the understanding of this parameter.
- **Dimension of the problem:** This parameter is a direct link with the problem to be optimized, it represents the space of research and evolution of the particles, and of course the dimension of the problem impacts the convergence of the algorithm, at the level of calculation time: dealing with a 2-dimensional problem does not take the same time as a 30-dimensional problem, and also at the level of the reliability of the results: the precision of the small-dimensional results is not the same as that linked to the problems with high dimension.
- **Particle arrangement:** Before starting the algorithm, the positions of the particles as well as their initial velocities must be initialized randomly according to a uniform law on [0..1]. This initial arrangement has an influence on the next displacement of each particle and therefore the convergence of the algorithm, especially in the case of the constitution of geographical neighborhoods. However, there is a set of automatic position generators, allowing different positions to be assigned to the whole swarm. Several studies exist in this direction [11]. The SOBOL sequence generator is one of the most efficient in the field, for a homogeneous arrangement of particles in a space of dimension n following a study made by [12] who used a certain sequence with small deviation to initialize the particles. The researchers used Halton, Sobol and Faure sequence generators to initialize the swarm. They then tested the offered variants using six standard test functions. They found that the performance of PSO with Sobol initialization is the best among all the techniques.
- **Stopping criteria:** The stopping criterion represents one of the key success of the PSO algorithm, choosing an optimal stopping criterion is not an easy task, and is not done randomly, but must be the result of a depth study of the problem and a massive experiment. This parameter differs according to the optimization problem and the constraints defined by the user, it is strongly advised to endow the algorithm with an exit gate since convergence towards the optimal solution is not guaranteed in all even if the experiments indicate the great performance of the method. As a result, several studies have been carried out in this direction [13], different propositions have taken place: the algorithm must then be executed as long as one of the convergence criteria has not been reached, this can be: the number maximum iterations; the global optimum is known a priori, one can define an "acceptable precision", the variation of the speed is close to 0. Other stopping criteria can be used according to the problem of

optimization posed and of the constraints defined by the user.

### D. Neighborhood in PSO

The neighborhood constitutes the structure of the social network. The neighborhood of a particle represents with whom each of the particles will be able to communicate. There are two main types of neighborhoods:

Geographic neighborhood: this type of neighborhood represents geographic proximity, it is the most natural notion of neighborhood for particle swarms, neighbors are considered the closest particles. However, with each iteration, the new neighbors must be recalculated from a predefined distance in the search space. It is therefore a dynamic neighborhood that must be defined and updated at each iteration.

Social neighborhood: this type of neighborhood represents social proximity, neighborhoods are no longer the expression of distance but the expression of the exchange of information, they are defined at initialization and are not modified by the following. Once the network of social connections is established, there is no need to update it. Therefore, it is a static neighborhood.

- Neighborhood topology

The network of relationships between all the particles is known as the "swarm topology". The choice of a neighborhood topology is very important, several topology studies have been carried out in this regard [14], different combinations have been proposed, the most used of which are mentioned above [2]:

Ring topology (Fig. 2 (a)): each particle is connected to n particles, (usually n = 3), each particle tends to point towards the best in its local neighborhood.

Radius topology (Fig. 2 (b)): communication between particles is made via a central particle, only the latter adjusts its position towards the best, if there is improvement in its position, the information is then spread to its congeners.

Star topology (Fig. 2 (c)): each particle is connected to all the others, the social network is complete, i.e. The neighborhood optimum is the global optimum.

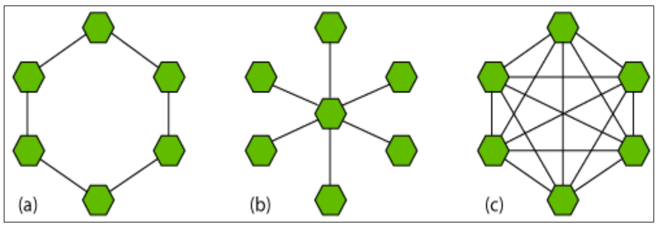

Fig. 2. Neighborhood topologies

## III. PARALLEL MODELS BASED ON THE PSO ALGORITHM

The PSO algorithm quickly took its place among the most powerful methods in the family of meta-heuristics dedicated to solve complex optimization problems. Although this meta-heuristic is very popular due to its robustness, it nevertheless presents several drawbacks, the most studied of which are the high computational time and premature convergence.

Parallelism is one of the concepts, which interested researchers in the field to remedy these drawbacks of the PSO method.

### A. Related work

As mentioned previously, in the implementation of the classical algorithm of the PSO method, all the calculations are done in a sequential manner; this is where the idea of parallelization arises in order to improve the performance of the algorithm. Several scenarios are proposed, we distinguish:

In [15], the authors tested in 2006 in both synchronous and asynchronous PSO parallel algorithms for optimization of typical parameters of the wings of a transport plane. The result indicates that the asynchronous PSO algorithm performs better than the synchronous PSO in terms of parallel efficiency. [16] Implemented in 2006 an asynchronous parallel PSO algorithm for analytical and biomechanical testing problems. The experimental results obtained show that the asynchronous PSO is 3.5 times faster than the synchronous PSO algorithm.

In [17] in 2007, a parallel PSO approach named (MRPSO) based on the MapReduce parallel programming model, with the aim of addressing complex optimization problems. MRPSO has been applied to a set of test functions well known in the optimization field for their difficulty. Based on the results obtained, MRPSO can handle up to 256 processors for moderately difficult optimization problems and tolerates node failure.

In this study [18] carried out in 2010, two algorithms are developed for determining the pricing of options using particle swarm optimization. The first algorithm we developed is the Synchronous Option Valuation Algorithm Using PSO (SPSO), and the second is the Parallel Synchronous Valuation Algorithm. The pricing results obtained from these two algorithms are close compared to the classic Black-Scholes-Merton model for simple European options. A test of the synchronous parallel PSO algorithm in three architectures was performed on a shared memory machine using OpenMP, a distributed memory machine using MPI, and a homogeneous multicore architecture running MPI and OpenMP (hybrid model). The results show that the hybrid model handles the charge well when there is an increase in the number of particles in simulation while maintaining equivalent precision.

A parallel particle swarm optimization algorithm is described in [4], proposed in 2012 to solve the problem of coverage of pursuit-evasion games, where several pursuers must cooperate to cover the potential flight zone of an agile fraudster within a reasonable time. The area to be covered is complex and therefore difficult to calculate analytically. With the use of the proposed parallel PSO algorithm, maximum coverage is achieved in less time, considering the minimum number of pursuers. Calculation time can be further reduced by optimizing the fitness function according to the locality of the data. In addition, the use of a variable length of the communication data frame makes it possible to reduce the communication time between processes when the number of processors increases (more than four in the test example). The simulation results show a comparison between the acceleration, the computation time before and after the optimization of the fitness function and the communication time between fixed and variable data frames. The positions and orientations of the prosecutors are also presented to show the effectiveness of the proposed parallel algorithm.

In [5], the authors in 2014 introduce several parallel functional skeletons, which, in a sequential implementation of the PSO method, automatically provide the corresponding parallel implementations. They use these skeletons and report some experimental results. They find that, despite the low effort required by programmers to use these skeletons, their empirical results show that the proposed skeletons achieve reasonable acceleration speeds.

In 2015, the authors of [19], developed the problem of Constraint Satisfaction Problems CSP which occur in different domains. Several methods are used to solve them. In particular, the PSO meta-heuristic, which effectively solves CSPs by drastically reducing the computational time needed to explore the solution search space. However, PSO is excessively expensive in the face of large instances. For this work, a particular interest was brought to the problems of satisfaction of maximum constraint (Max-CSP) by proposing a new approach of resolution, which allows solving efficiently Max-CSP, even with large instances. The goal was to implement a PSO-based method using the Graphics Processing Unit (GPU) architecture as a parallel computing framework. Two parallel models are offered; the first is a GPU parallel PSO for Max-CSP (GPU-PSO) and the second is a GPU distributed PSO for Max-CSP (GPU-DPSO). The experimental results show the efficiency of the two proposed approaches and their ability to exploit the GPU architecture.

In [6] two parallel strategies are proposed in 2019, based on several swarms to solve multi-objective optimization problems. The multiple swarms co-evolving in parallel and interacting through migration. Different policies for triggering migration are proposed and evaluated. An in-depth experimental evaluation of the algorithms is presented, as well as a study of the impact of the proposed methods on the convergence and diversity of research in many scenarios of multi-objective optimization. The first strategy is based on Pareto domination and the other on decomposition. Several swarms run on independent processors and communicate in broadcast over a fully connected network. A study of the impact of using synchronous and asynchronous communication strategies for the decomposition-based approach. Experimental results have been obtained for several reference problems. The conclusion was that parallelization has a positive effect on the convergence and diversity of the optimization process for multi-objective problems. However, there is no single strategy that works best for all categories of problems. In terms of scalability, for higher numbers of goals, parallel algorithms based on decomposition always show the best results.

## B. *Proposed parallel model*

The scenario that we have adopted in this proposed model called PN-PSO (Parallel Neighborhood PSO) allows parallelizing the calculations by launching a set of threads on batches of particles positioned in different neighborhoods. Threads, (a sort of Java processes in our experiment), run in parallel with each iteration of the algorithm.

Each thread performs the processing of an iteration of its batch of particles, and waits for the other threads to complete their processing to update the neighborhoods and start a new iteration. This scenario is repeated until a satisfactory solution is obtained "reaching the stopping criterion".

The particularity of this model consists in taking advantage of the robustness of the PSO algorithm in the choice of the right parameterization in order to create diversity in the search (in our case: the distribution of particles in the search space and our notion of neighborhood) and in information sharing to facilitate convergence. Parallel computing speeds up calculations in order to have an "optimal" solution in an optimized computing time. Fig. 3 is a representation of the proposed approach [20].

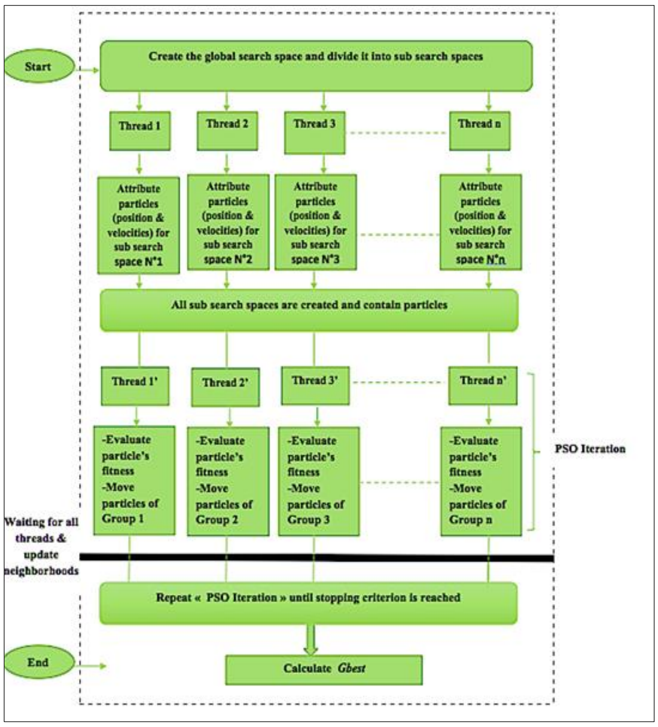
Start
Create the global search space and divide it into sub search spaces
Thread 1
Thread 2
Thread 3
Thread n
Attribute particles (position & velocities) for sub search space N°1
Attribute particles (position & velocities) for sub search space N°2
Attribute particles (position & velocities) for sub search space N°3
Attribute particles (position & velocities) for sub search space N°n
All sub search spaces are created and contain particles
Thread 1'
Thread 2'
Thread 3'
Thread n'
-Evaluate particle's fitness -Move particles of Group 1
-Evaluate particle's fitness -Move particles of Group 2
-Evaluate particle's fitness -Move particles of Group 3
-Evaluate particle's fitness -Move particles of Group n
PSO Iteration
Waiting for all threads & update neighborhoods
Repeat « PSO Iteration » until stopping criterion is reached
End
Calculate Gbest

Fig. 3. The proposed Parallel PSO model

- Algorithm Framework

The main steps of the PN-PSO algorithm are as follows:

*Step 1: Create the global search space and divide it into sub spaces according to a value of step.*

*Step 2: Generate randomly a set of particles by attributing their positions and speeds.*

*Step 3: Each sub set of particles 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: Wait for all threads and update neighborhoods.*

*Step 6: If the stopping criterion is satisfied, stop, otherwise go to step 4.*

## IV. EXPERIMENT AND RESULATS

This section deals with a description of the experiments and an analysis of the obtained results of the proposed model based on the PSO algorithm.

### A. *Description of experiments*

The modification of the basic algorithm of PSO method for our two models involves three essential points: the notion of neighborhood, parameter adaptation, and parallel computing. These changes to the algorithm improve its performance.

Our algorithm was programmed in JAVA 1.8, and the experiments were done on a MacBook Pro OS X 10.13.15, Core i7, 16 GB machine. Threads are the technology used in Java to multitask applications. They share the same memory, as well as resources (in memory), for this, the threads risk competing and corrupting the system.

This is where concurrent programming comes in, bringing together a set of features and techniques to enable synchronization of tasks running in parallel. Java manages processes better than threads, but threads are used a lot more because they are better integrated into the Java language and less memory intensive. While Java is a robust language with many advantages (portability, inheritance, etc.), but especially to the concept of concurrent programming (parallelism and synchronization) we opted for this language in order to carry out our experiments and take advantage of the advantages of parallelism in terms of reducing the computation time, and also to make the most of the hardware resources of the machine.

To test our proposed parallel model, a set of ten test functions has been selected (see Table. 1) to evaluate the performance of the proposed model.

TABLE I. DESCRIPTION OF THE USED FUNCTIONS

| Function | Range | $f_{min}$ |
|---|---|---|
| **$f_1$ Rosenbrock** | ±30 | 0 |
| **$f_2$ Himmelblau** | ±30 | -3.78396 |
| **$f_3$ Beale's** | ±4.5 | 0 |
| **$f_4$ Easom** | ±100 | -1 |
| **$f_5$ McCormick** | ±4.0 | -1.9133 |
| **$f_6$ Three-hump camel** | ±5.0 | 0 |
| **$f_7$ Hölder table** | ±10 | -19.2085 |
| **$f_8$ Matyas** | ±10 | 0 |
| **$f_9$ Booth's** | ±10 | 0 |
| **$f_{10}$ Goldsteinprice** | ±2.0 | 3 |

In the PSO algorithm each parameter has an important influence on the behavior of the particles and therefore on the convergence of the algorithm; and even if the PSO method presents satisfactory results, the choice of the right parameterization of the method remains a critical point as well as one of the keys to success for any PSO algorithm. In the descriptive section of the PSO method, we have presented some parameters that influence the behavior of particles in their movements in search of the optimum.

The parameters that we developed in our model is the use of several variable parameters that can be modified from the user interface dedicated for it. It all depends on the requirements of the optimization problem. Massive experimentation was carried out to find the appropriate set of parameters; it has given results, which we consider satisfactory.

It is important to note that a simple change in the value of a parameter can greatly change the result, and can even lead to premature convergence. For the present study, which deals with the moderate size, two-dimensional problems, the list of parameters, which gave sufficient good results are mentioned in the table below (see Table. 2).

TABLE II. DESCRIPTION OF THE USED PSO PARAMETERS

| PSO parameters | Parallel model |
|---|---|
| **Swarm size** | 30 |
| **Number of iterations** | 50-80 |
| **Acceleration coefficients** | C1 = 1.25, C2=2.25, C3=2.25 |
| **Inertia factor** | (0.4 – 0.2) |
| **Communication topology** | Ring |
| **Number of threads** | Depends on objective function |
| **Stopping criteria** | - The maximum number of iterations.<br>- The maximum number of iterations without improvement of Gbest. |

### B. *Results*

The table below shows the results details of the average of 1000 executions: the values of execution time in milliseconds, the SR: the success rate which is the percentage of convergence of the function towards the right solution, the EvalIF which represents the number of evaluation of the objective function, and this for the basic PSO and the proposed parallel model on a set of ten functions.

TABLE III. EXPERIMENTAL RESULTS

| | PN-PSO model | | | Basic PSO | | |
|---|---|---|---|---|---|---|
| Funtions | SR % | Time (ms) | EvalF | SR% | Time (ms) | EvalF |
| F1 | 100 | 891.2 | 300123 | 99 | 967.1 | 301213.1 |
| F2 | 100 | 15.4 | 9702 .5 | 95 | 23.7 | 11089.3 |
| F3 | 100 | 511.3 | 21760.2 | 100 | 702.4 | 23099.4 |
| F4 | 100 | 10.3 | 6222.2 | 100 | 12.1 | 7680.1 |
| F5 | 100 | 702.9 | 26201.3 | 100 | 911.4 | 28901.3 |
| F6 | 100 | 831.6 | 279008 | 100 | 997.9 | 384098 |
| F7 | 100 | 12.8 | 7907.1 | 100 | 14.3 | 8709.8 |
| F8 | 100 | 345.5 | 120312 | 100 | 456.7 | 133098 |
| F9 | 100 | 9.8 | 5780.3 | 100 | 11.6 | 7002 |
| F10 | 100 | 13.4 | 8872 | 100 | 15.6 | 9995.4 |

From the results obtained, we have observed the performance of the PN-PSO algorithm in terms of the solution quality; it avoids the convergence of particles in local optima. The computation time in the proposed PN-PSO model is also lower than the sequential model.

## V. CONCLUSIONS

This paper presents a parallel model based on the Particle Swarm Optimization meta-heuristic. The objective was to propose solutions to the two drawbacks of the method: premature convergence and the high computing time. In the literature, several improving versions of the PSO method are proposed either by adding new parameters, by parallelizing it or by hybridizing it with other meta-heuristics.

The proposed model that we have presented in this paper is based on two concepts: parallelization and neighborhood. The combination of these two notions improved the performance of the method in terms of quality of the solution and of computation time. Our PN-PSO model uses the notion of dynamic neighborhood, which allows to create diversity in the search as well as a better exploration of the search space in order to improve the quality of the solution and avoid the stagnation of the algorithm in a local optimum; parallel computing is used to speed up calculations in order to have an "optimal" solution in a reduced computing time.

## REFERENCES

[1] Y. Cooren, Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique, Doctoral thesis, Paris 12 Val de Marne University, France, 2008.

[2] N. Elhami, Contribution aux méthodes hybrides d'optimisation heuristiques: Distribution et application à l'interopérabilité des systèmes d'information", Doctoral thesis, Mohammed V University Rabat, Morocco & Normandy University, France, 2013.

[3] K. Byung-I et G. Alan, Parallel asynchronous particle swarm optimizationm, in: International Journal For Numerical Methods In Engineering, vol. 67, pp. 578-595, 2006.

[4] Shiyuan Jin, Damian Dechev, Zhihua Qu, Parallel Particle Swarm Optimization (PPSO) on the Coverage Problem in Pursuit-Evasion Games, in : Conference: Proposed for presentation at the 20th High Performance Computing Symposium (HPC 2012), Orlando, FL. 2012.

[5] P. Rabanal, I. Rodríguez et F. Rubio, Parallelizing Particle Swarm Optimization in a Functional Programming Environment, Journal of Algorithms2014 : vol. 7, pp. 554–581, 2014.

[6] Arionde Campo Aurora T.R.Pozo Elias P.Duarte, Parallel multi-swarm PSO strategies for solving many objective optimization problems. Journal of Parallel and Distributed Computing. Vol 126, pp.13-33, 2019.

[7] J. Kennedy et 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.

[8] K. E. Parsopoulos et M. N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing: an international journal, 1(2-3), pp. 235-306, 2002.

[9] M. E. Hyass et P. Hyass, Good Parameters for Particle Swarm Optimization, Laboratories Technical Report no. HL1001. 2010.

[10] M. Zemzami, VARIATIONS SUR PSO : APPROCHES PARALLELES, JEUX DE VOISINAGES ET APPLICATIONS., Doctoral thesis, Ibn Tofail University Kenitra, Morocco & Normandy University, France, 2019.

[11] Pant, M., Thangaraj, R., & Abraham, A, Particle swarm optimization using adaptive mutation, in: 19th International Conference on Database and Expert Systems , Washington, DC, USA, pp. 519-523, 2008.

[12] Nguyen, U. Q., Hoai, N. X., McKay, R., & Tuan, P. M, Initializing PSO with randomized low-discrepancy sequences: the comparative results, in: Proceedings of the IEEE Congress on Evolutionary Computation pp. 1985-1992, 2007.

[13] K. Zielinski et R. Laur, Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimizationm Advanced in Differential Evolution, the series Studies in Computational Intelligence Vol. 143, pp. 111-138 Springer, Berlin Heidelberg. 2008.

[14] R. Mendes, Population Topologies and Their Influence in Particle Swarm Performance,. Doctoral thesis, Minho Uviersity, Portugal, 2004.

[15] Gerhard Venter et Jaroslaw Sobieszcanski, Parallel particle swarm optimization algorithm accelerated by asynchronous evaluations, J. Aerosp. Comput. Inf. Commun. 3(3), pp. 123–137, 2006.

[16] Byung-II Koh, Alan D. George, Raphael Haftka et Benjamin J Fregly, Parallel asynchronous particle swarm optimization, Communications in Numerical Methods in Engineering 67(4) pp. 578-595, 2006.

[17] Andrew W. McNabb ; Christopher K. Monson ; Kevin D. Seppi, Parallel PSO using MapReducem in : IEEE Congress on Evolutionary Computation. 2007.

[18] Hari Prasain ; Girish Kumar Jha ; Parimala Thulasiraman ; Ruppa Thulasiram, A parallel Particle swarm optimization algorithm for option pricing, in IEEE: International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). 2010.

[19] Narjiss Dali et Sadok Bouamama, GPU-PSO : Parallel Particle Swarm Optimization approaches on Graphical Processing Unit for Constraint Reasoning: Case of Max-CSPs, in: 19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems. Procedia Computer Science 60 (2015) pp.1070 – 1080, 2015.

[20] M.Zemzami, N.Elhami, M.Itmi et N.Hmina, A New Parallel Approach For The Exploitation Of The Search Space Based On PSO Algorithm, in: 4th International Colloquium in Information Science and Technology (CIST'16). Tangier. Morocco. 2016.