Int. J. Simul. Multidisci. Des. Optim. **11**, 5 (2020)

https://doi.org/10.1051/smdo/2019022

IJSMDO
Available online at:
https://www.ijsmdo.org

REVIEW OPEN ACCESS

A comparative study of three new parallel models based on the PSO algorithm

Maria Zemzami[1,2,*], Norelislam El Hami[1], Mhamed Itmi[2], and Nabil Hmina[1]

[1] LGS Lab – ENSA, Ibn Tofail University, Kenitra, Morocco
[2] LITIS Lab – INSA, Normandy University, Rouen, France

Received: 15 April 2019 / Accepted: 2 December 2019

Abstract. Meta-heuristic PSO has limits, such as premature convergence and high running time, especially for complex optimization problems. In this paper, a description of three parallel models based on the PSO algorithm is developed, on the basis of combining two concepts: parallelism and neighborhood, which are designed according to three different approaches in order to avoid the two disadvantages of the PSO algorithm. The third model, SPM (Spherical-neighborhood Parallel Model), is designed to improve the obtained results from the two parallel NPM (Neighborhood Parallel Model) and MPM (Multi-PSO Parallel Model) models. The experimental results presented in this paper show that SPM model performed much better than both NPM and MPM models in terms of computing time and solution quality.

Keywords: Optimization / metaheuristic / PSO / SA / parallelization / computing time

1 Introduction

Solving different kinds of problems in our daily lives has led researchers to propose resolution methods and to make great efforts to improve their performance in terms of the necessary computing time and/ or the quality of the proposed solution. PSO is one of the most powerful methods in the field of optimization, and has been applied to a large number of real optimization problems.

In literature, there are many methods to solve optimization problems. Optimization methods are often classified into two classes: the class of exact methods and the class of the approximate methods. The exact methods guarantee the optimality of the solution; however, they are often expensive in terms of computing time, namely in solving large optimization problems. Therefore, the approximate methods are opted for as an alternative of the exact methods. Though the latter do not guarantee the optimality of the solution, they are less expensive than the former in terms of necessary computing time. In fact, the approximate methods allow the resolution of optimization problems of different sizes in a reasonable computing time.

The investigation in the field of approximate methods has given the emergence of another class of methods called "Metaheuristics". Metaheuristics are general methods applicable on a wide range of optimization problems, often, inspired by natural systems in different fields: physics, biology, ethology ... etc.

The idea of taking inspiration from natural systems to propose methods for solving optimization problems has given rise to a subclass of metaheuristics, they are based on intelligence by swarm called "Swarm Intelligence". Particle Swarm Optimization (hereafter, PSO) is the main metaheuristic in this subclass; it is a population-based metaheuristic inspired by an analogy with ethology.

The PSO was proposed in 1995 by James Kennedy (psychologist) and Russel Eberhart (electrical engineer) [1] for solving continuous optimization problems. Initially, J. Kennedy and R. Eberhart sought to simulate the ability of birds to fly synchronously and to change suddenly direction while remaining in optimal training. The model they proposed was, then, extended into an efficient optimization algorithm.

The simplicity and the performance of this method have attracted interest from several communities of researchers who conducted optimization studies and applied this metaheuristic for solving several continuous and/or discrete optimization problems. As a result, several alternatives to the original PSO algorithm have been proposed in the literature to improve its performance for solving different problems [2–6]. In the present study a comparison of three parallel approaches based on PSO algorithm is considered.

The present paper has four sections including the introduction. In the next section, a brief description of the

* e-mail: maria.zemzami@gmail.com

basic PSO will be given along with a discussion of the three parallel models. Section 3 will describe the experimental settings and analyze results, respectively. Finally, Section 4 will handle the synthesis and conclusion.

2 Parallel optimization approaches

The performance of multiple optimization algorithms is affected when solving complex optimization problems; PSO is one of them. This is what has, in fact, motivated the development of parallel models.

2.1 Particle swarm optimization (PSO)

The PSO method is inspired by the social behavior of animals living in swarms, such as insects, fish or birds in search for food. The global intelligence of the swarm is, therefore, the direct result of local interactions between the different particles of the swarm. The performance of the whole system is greater than the sum of the performances of its parts. Kennedy and Eberhart were inspired by these social behaviors to create the PSO algorithm.

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 uses cooperation between individuals.

2.1.1 Formalization

The particle i will move between the iterations t and $t+1$, depending on its speed and the two best positions it knows (i.e. its own best position and the best position in the swarm) according to the two following equations [1]:

$$\begin{cases} V_{iD(t+1)} = V_{iD(t)} + C_1 r_1 (Pb_{iD(t)} - X_{iD(t)}) \\ \qquad\qquad + C_2 r_2 (Pg_{iD(t)} - X_{iD(t)}) \qquad (1) \\ X_{iD(t+1)} = X_{iD(t)} + V_{iD(t)} \qquad (2) \end{cases}$$

where $X_{iD(t)}$, $X_{iD(t+1)}$: the position of the particle i in the dimension D at times t and $t-1$, respectively. $V_{iD(t)}$, $V_{iD(t+1)}$: the velocity of the particle i in the dimension D at times t and $t-1$, respectively. $Pb_{iD(t)}$: the best position obtained by the particle i in dimension D at time $t-1$. $Pg_{iD(t)}$: the best position obtained by the swarm in dimension D at time $t-1$. C_1 and C_2: two constants that represent the acceleration coefficients. r_1 and r_2: random numbers drawn from the interval [0,1].

Each particle moves around the global search space taking into consideration the three terms mentioned above in the movement equations (1) and (2) (see Fig. 1).

2.1.2 PSO algorithm

The PSO algorithm is a stochastic process invented by [1]. It starts with initializing the swarm size and the different PSO parameters, assigning to each particle an initial position and velocity. Then, it calculates the fitness of the particles in order to find the best position in the swarm. At each iteration of the research process, the particles move according to equations (1) and (2). Their fitness is calculated in each time; therefore, Pb and Pg are updated accordingly. The process is repeated until the stopping criterion is met. Below is the basic PSO flowchart (see Fig. 2).

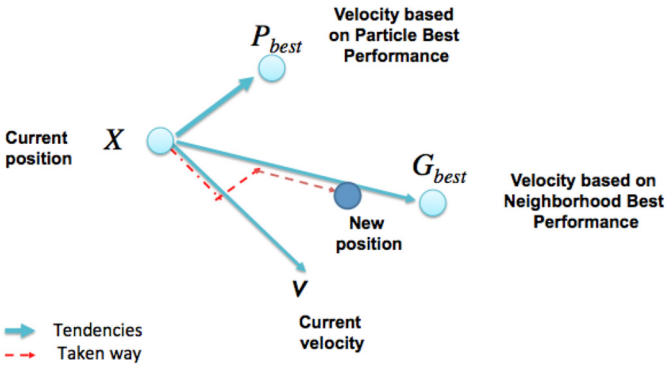

Fig. 1. The particle movement by basic PSO (proposed by [7]).

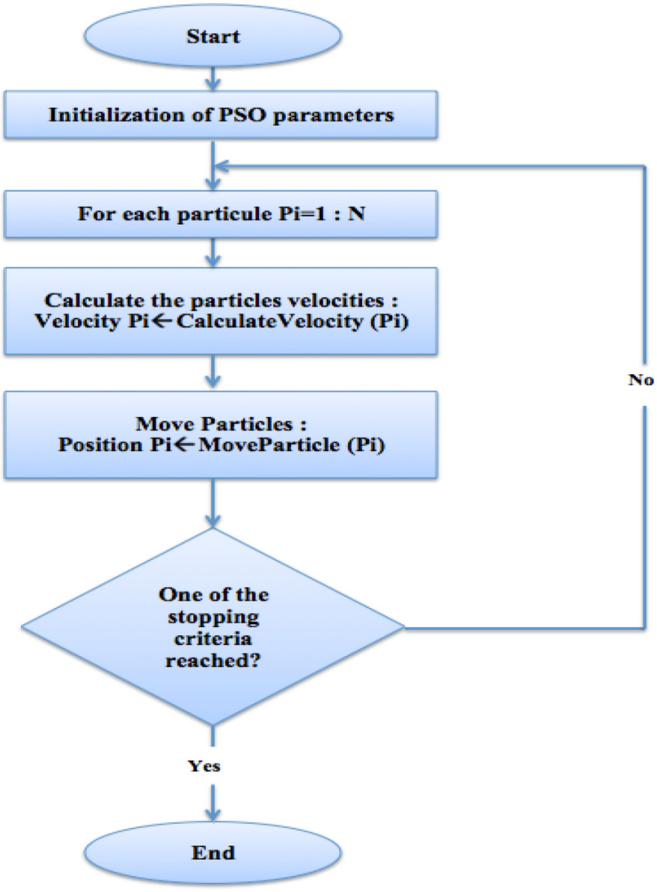

Fig. 2. The basic PSO algorithm flowchart.

2.2 The proposed parallel models

The performance of the PSO method is affected when dealing with complex optimization problems; this motived

the use of parallelization to improve the performance of the PSO algorithm. Several authors have been interested in doing so and proposed parallel models based on the PSO method [8–11]. The three parallel models are developed as follows:

– 1st Model: Neighborhood Parallel Model (NPM).
– 2nd Model: Multi-PSO Parallel Model (MPM).
– 3rd Model: Spherical-neighborhood Parallel Model (SPM).

Although the PSO meta-heuristic is declared one of the most efficient methods in the field of optimization, it has some disadvantages, namely, premature convergence and high running time. These are the most studied in this field:

– Premature convergence: the execution structure and the principle of particles movement for the PSO algorithm lead to premature convergence.

Example: if a particle moves to a place "containing a suboptimal solution" and declares in this iteration the best of its group, this information will affect the movement of all the other particles during the next iteration, therefore, the whole swarm will go towards the place in question. This will generate a premature convergence.

– High running time: all calculations in the PSO algorithm are done sequentially. This can lead to considerable computation time, especially for complex optimization problems.

Example: if we launch the PSO algorithm on a population of particles "higher dimension", calculations will be done at each iteration for each particle for all its dimensions in a sequential way. This will cause a high running time.

2.2.1 Neighborhood Parallel Model (NPM)

In this first model named NPM, a parallelization based on the PSO algorithm is developed. Two concepts are combined in NPM approach: parallel computing and dynamic neighborhood. The NPM algorithm starts by a division of the global search space into sub-spaces (which will represent the particles' neighborhoods), and, then, a random initialization of the particles in the search space is made.

Threads, that are Java processes, are running in parallel with each iteration of the algorithm. Each thread executes the PSO processing for a set of particles positioned in different neighborhoods. The particles of each subspace constitute a group (neighborhood), and at each iteration, the particles move around the search space looking for the optimum. In the next iteration, they automatically change their positions and also their neighborhoods. Synchronization is performed at the end of each iteration in order to update neighborhoods and start a new iteration. This scenario is repeated until the end of the algorithm (when the stopping criterion is met).

For more details about this model, the reader is referred to [12]. The proposed NPM algorithm is represented in the figure below (see Fig. 3).

2.2.1.1 Algorithm framework

The main steps of NPM 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.

2.2.1.2 NPM model analysis

The NPM model, as any algorithm, has some disadvantages. We distinguish:

– Stochastic distribution of particles in the search space: in the initialization of the NPM algorithm, the random disposition of particles in the search space can lead to premature convergence. Since it is possible to have portions of the search space not exploited, these portions can contain the optimal solution. In addition, as the search of the optimum is done in a group (which changes at each iteration), the particles share their travel information at each iteration, which can lead to a stagnation of the algorithm in a local optimum.
– Synchronization: since each thread deals with the processing of a set of particles located in different neighborhoods, synchronization is needed at each iteration in order to allow the sharing of the movement information between particles before the next iteration; this is what generates losses of time at each iteration of the algorithm.

Although the experimental results of the proposed NPM parallel model are better than those of the sequential model in terms of the solution quality and time consuming [12], the NPM model has its disadvantages. This led us to design a new model to avoid the disadvantages of NPM.

2.2.2 Multi-PSO Parallel Model (MPM)

The MPM model is designed to improve the obtained results from the NPM model by avoiding its disadvantages. The MPM model is based on a set of PSO algorithms launched in parallel in search of the optimum. The global search space is divided into subspaces using a value of "step" attributed to each axis of the search space (depending on the dimension of the objective function). Then, we allocate to each subspace a group of particles; with this we are sure that all portions of the search space will be explored and exploited. After dividing the global search space into sub-spaces, and initializing them by particles, the processing of each subspace is attributed to a "thread". Our model is based on parallel processing. In fact, processes are meant to perform calculations on sets of particles. Each set of particles is located in its subspace, thus, forming a group. There is no communication between particles of different sub-spaces; each group is looking for

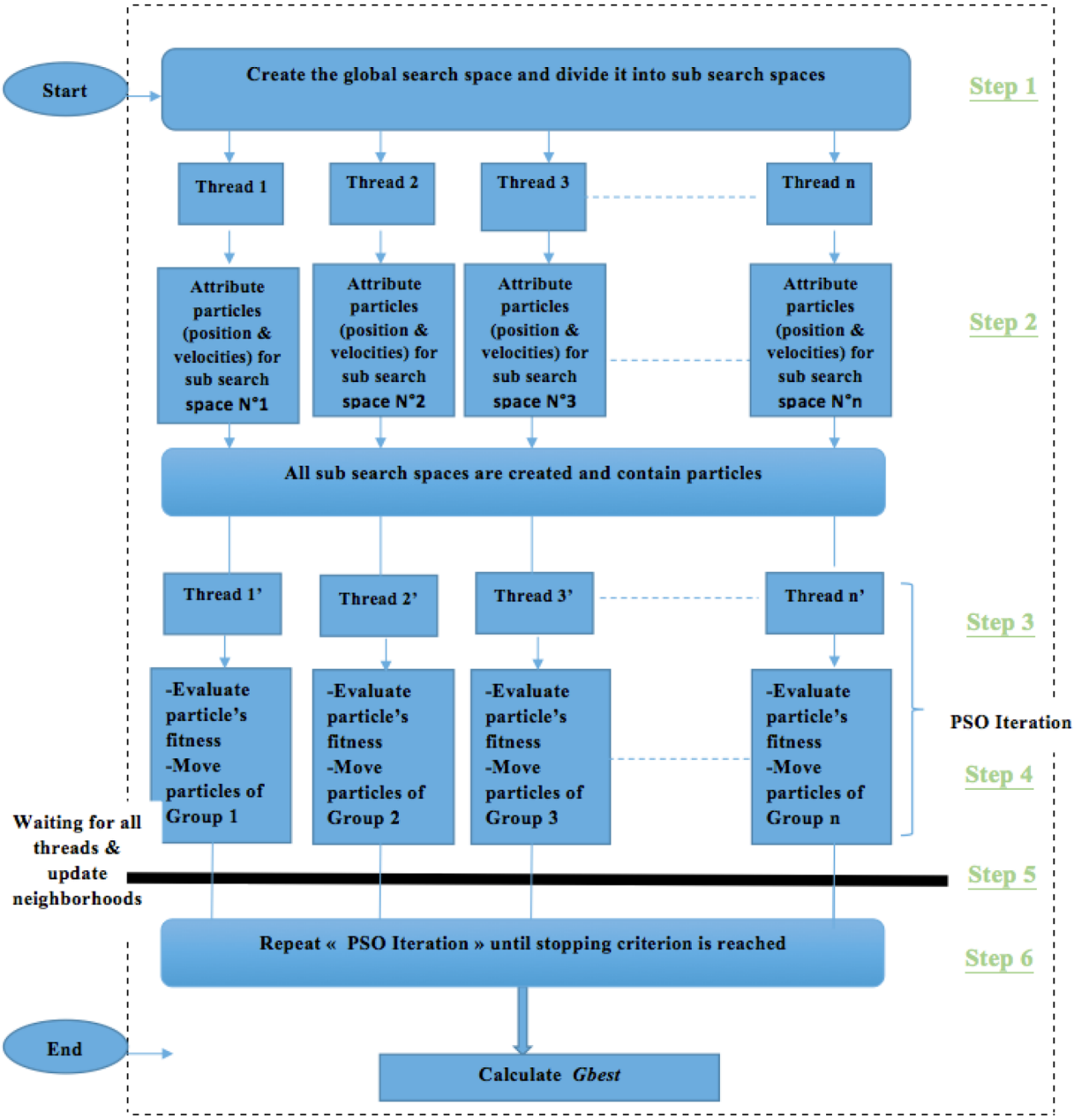

Fig. 3. The proposed NPM algorithm.

the optimal solution independently. At the end, when all threads finished their processing, a comparison of the obtained results of each thread is made to find the most optimal solution.

The exploitation and the exploration of the search space are two contradictory behaviors that work together to solve the problem. The right balance between them is integral to the performance of the PSO algorithm. The correlation of these behaviors with our approach is apparent, this was the inspiration behind the first part of MPM algorithm. The second part consists on minimizing computational costs by using parallel computation.

For more details about this model, the reader is referred to [7]. The proposed NPM algorithm is described in the figure below (see Fig. 4).

2.2.2.1 *Algorithm framework*

The main steps of MPM algorithm are as follows:

Step 1: Create the global search space and divide it into sub search spaces.

Step 2: Attribute particles for each sub search space by generating their positions, velocities and communication topology.

Step 3: Create PSO Thread* by sub search space.

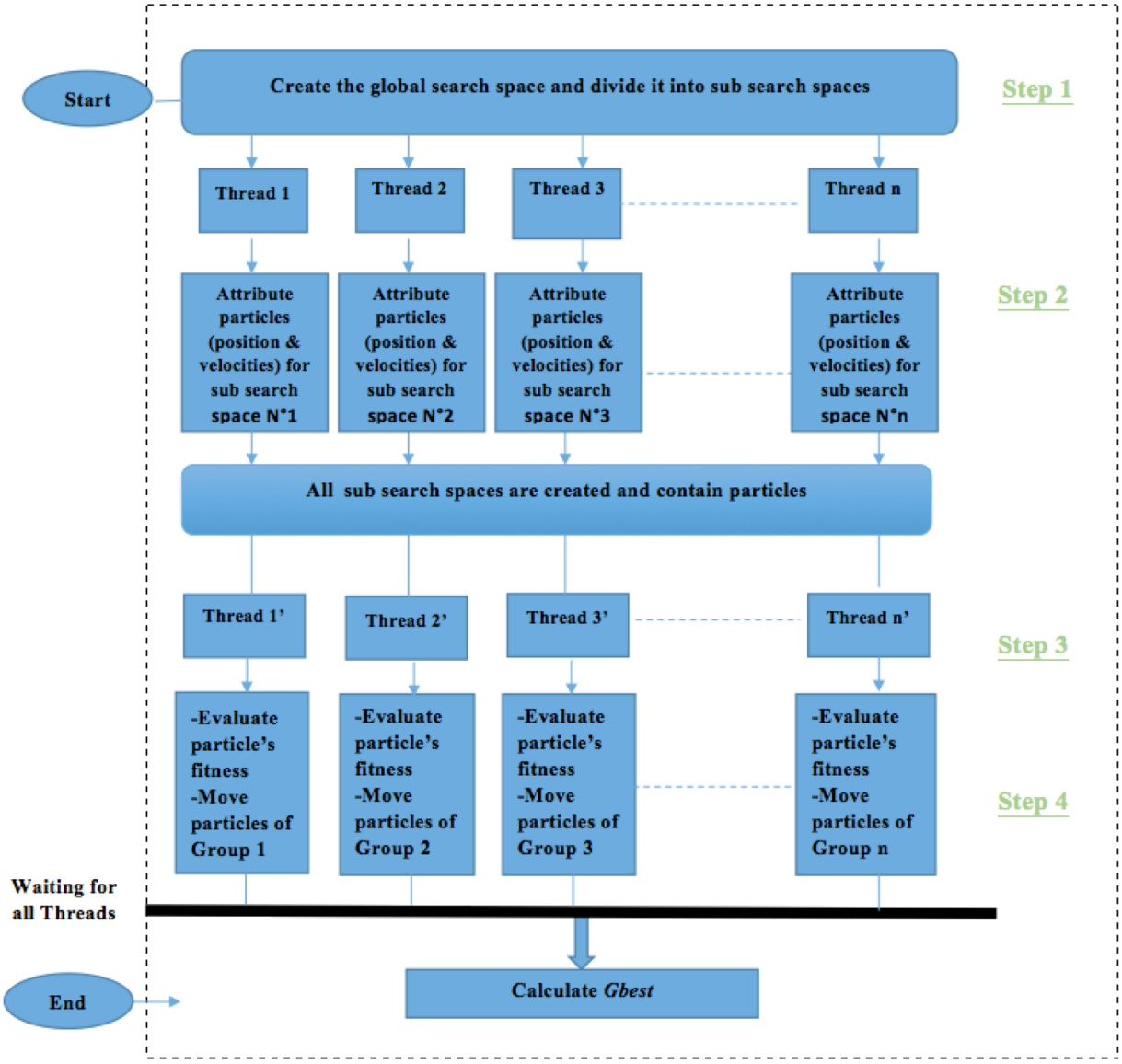

Fig. 4. The proposed MPM algorithm (proposed by [7]).

Step 4: Evaluate the best solution for each PSO Thread.

*PSO Thread steps:

Step 1: Evaluate each particle's fitness.

Step 2: For each particle, if its fitness is smaller than its previous best (Pb) fitness, update Pb.

Step 3: For each particle, if its fitness is smaller than the best one (Pg) of all the particles, update Pg.

Step 4: Move all particles according to the formula (1) and (2).

Step 5: It the stopping criterion is satisfied, then stop, else go to Step 1.

2.2.2.2 MPM model analysis

To overcome the two disadvantages of the NPM model already mentioned, a uniform distribution of particles throughout the search space is made. In addition, to avoid synchronization, MPM model allows each group to look independently for the optimum without having to communicate at each iteration its movement information.

In fact, MPM model is efficient and gives good results, especially for small size optimization problems with many local optima, given that the whole search space is explored. The obtained results are detailed in [7]. However, this model is not adequate for large optimization problems (high dimension), because the division of the global search space and the creation of the threads is costly in terms of calculation time.

There are also very critical parameter choices to make for this MPM model: the stopping criterion, the step value (which represents the number of research areas), and the number of particles per group.

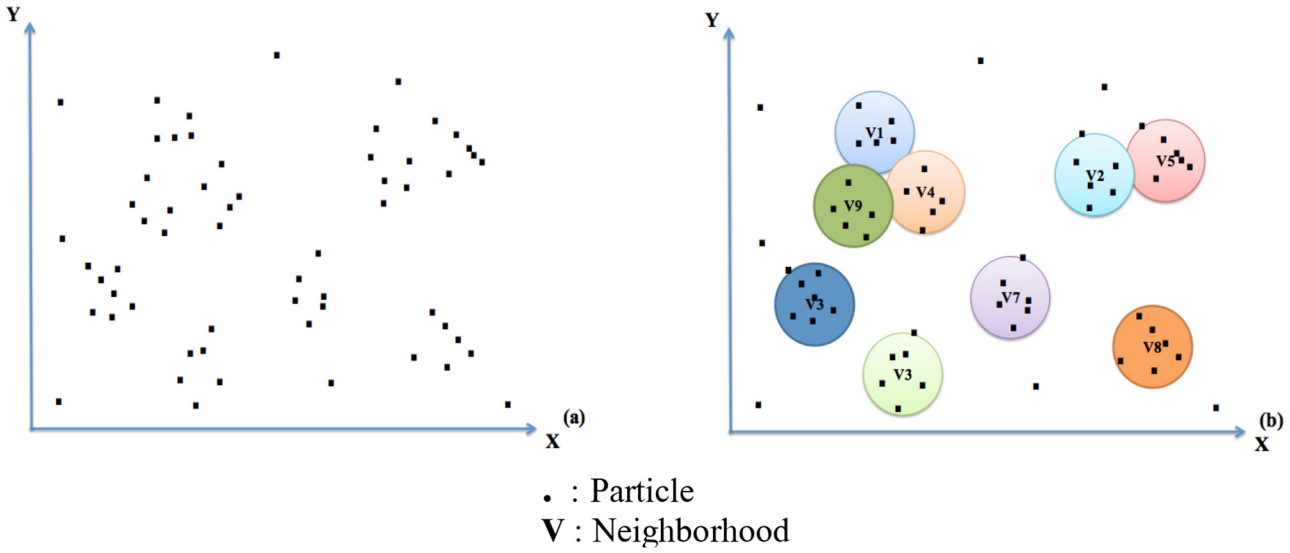

Fig. 5. An example of particle distribution in a two-dimensional search space at iteration t (a) and the representation of spherical neighborhoods at iteration $t + 1$ (b).

All the experiments and studies based on these two parallel NPM and MPM models led our research to the realization of a consistent parallel model, which will be intended for a wide variety of optimization problems, especially complex optimization ones.

2.2.3 Spherical-neighborhood Parallel Model (SPM)

To avoid the disadvantages of the two parallel models presented above, we have designed a new model named SPM. In this model, two concepts are associated: parallel computing and evolutionary neighborhood. This combination, according to our approach, improves the experimental results obtained from NPM and MPM models.

The idea of the model is to use groups of particles (dynamic neighborhoods) moving around the search space in search of the optimum; each group is looking for the solution independently of the other groups, and, at the end of each iteration, new groups are created that shared their travel information to start a new iteration.

The neighborhoods are created at the beginning of the program. They have the shape of large and small spheres according to the number of particles that compose them, and, which are updated at each iteration of the algorithm (see Fig. 5).

The processing of each sphere (neighborhood) is assigned to a thread. At each iteration, the number of neighborhoods is equal to the number of created threads.

The PSO processing of each neighborhood is independent; there is no sharing of the global best between groups. Each particle moves taking into consideration its best position and its neighborhood best position.

However, it's important not to consider the global best in the particle movement formula, because it allows a better exploration and exploitation of the search space without influencing particles by the result of the global best in each iteration of the algorithm, which can lead to premature convergence. Below is the flowchart of the proposed SPM model (see Fig. 6).

2.2.3.1 Algorithm framework

The main steps of SPM 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.

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

3 Description of our experiment and results

This section deals with a description of the experiments and an analysis of the obtained results of the NPM, MPM, and SPM models.

3.1 Benchmark problems

In the literature, there is a set of test functions used to evaluate the performance of the optimization algorithms. This series of problems are created specially to determine the efficiency of optimization methods, in terms of running time, success rate, accuracy, robustness, etc. Four classes are distinguished [13] (see Tab. 1).

In this paper, a set of ten test functions is used from the four classes (see Tab. 2). These functions have some properties similar to real-world problems and

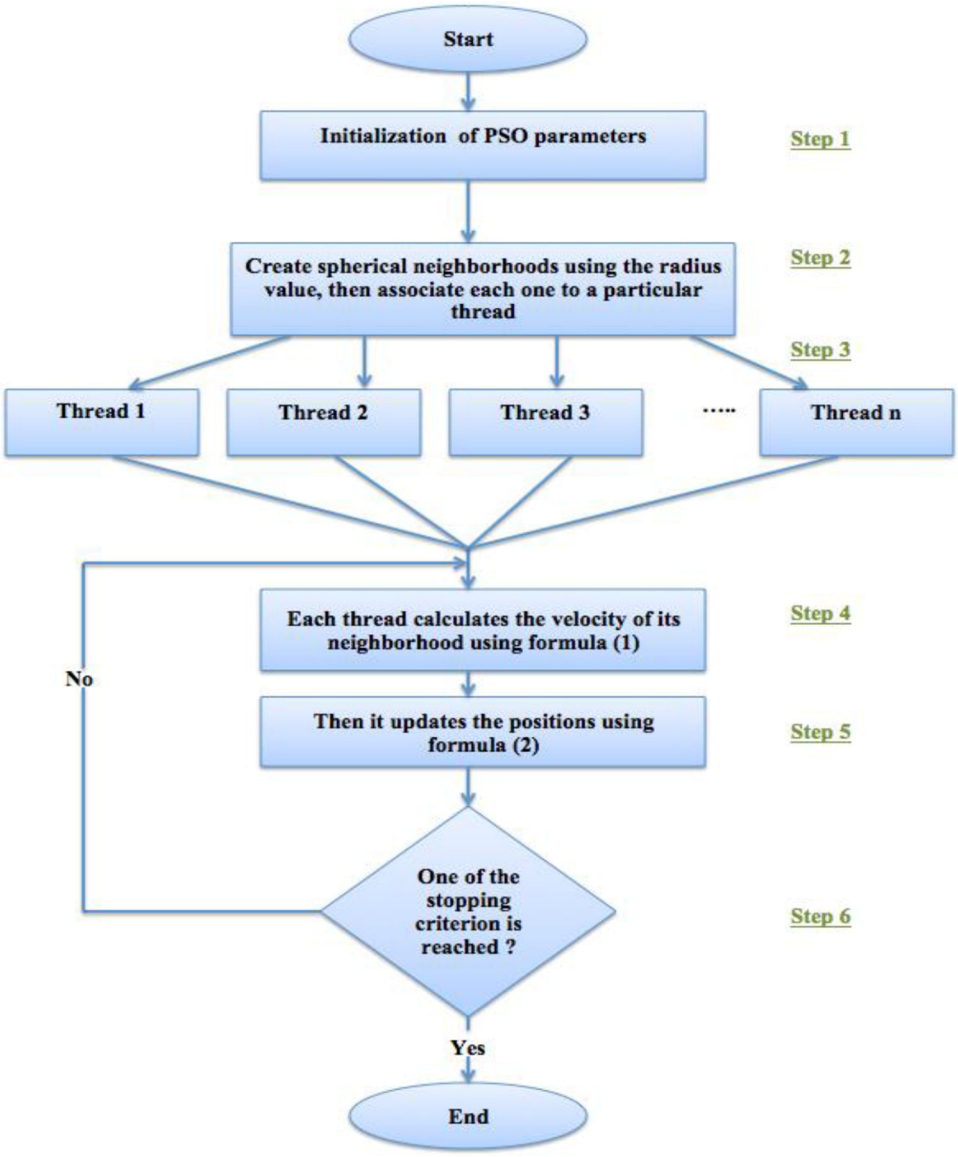

Fig. 6. SPM algorithm flowchart.

Table 1. Test functions classes.

Class	Type	Description
A	Unimodal	Convex, high dimension
B	Multimodal	Two dimensions with a small number of local extrema
C	Multimodal	Two dimensions with a large number of local extrema
D	Multimodal	High size, with a high number of local extrema

provide a good basis for testing the credibility of an optimization algorithm. All tests are formulated as minimization problems. They can also be used for maximization problems by simply reversing the sign of the function.

3.1.1 Experimental settings

The PSO method is composed of a set of parameters that must be defined at the initialization of the algorithm. The choice of the values of each parameter is very critical

Table 2. Description of the used functions in our experiments.

Function	Formula	Range	$f_{\min}$	Dim
f_1 Sphere	$f(x) = \sum_{i:1}^{n} x_i^2$	±5,12	0	30
f_2 Griewank	$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	±600	0	30
f_3 Rosenbrock	$f(x) = \sum_{i:1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	±30	0	30
f_4 Rastring	$f(x) = An + \sum_{i=1}^{n}[x_i^2 - A\ \cos(2\pi x_i)]$	±5.12	0	30
f_5 Easom	$f(x,\ y) = -\cos(x)\cos(y)\exp(-((x-\pi)^2 + (y-\pi)^2))$	±100	0	30
f_6 Ackley	$f(x,y) = -20\ \exp\ [-0.2\sqrt{0.5(x^2+y^2)}] - \exp\ [0.5(\cos 2\pi x + \cos 2\pi y)] + e + 20$	±5	0	30
f_7 Cross-in-tray	$f(x,y) = -0.0001\left[\left\lvert \sin x\ \sin\ y\ \exp\ \left(\left\lvert 100 - \frac{\sqrt{x^2+y^2}}{\pi}\right\rvert\right)\right\rvert + 1\right]^{0.1}$	±10	−2.06261	10
f_8 Schaffer	$f(x,y) = 0.5 + \frac{\sin^2(x^2+y^2) - 0.5}{[1 + 0.001(x^2+y^2)]^2}$	±100	0	10
f_9 Step	$f(x) = \sum_{i=0}^{n} \alpha_i \chi_{Ai}(\chi)$	±100	0	30
f_{10} Himmelblau	$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$	±30	−3.78396	2

because it influences the algorithm performance. A small change in a parameter value can influence the search, and greatly change the results. Choosing the right PSO settings is very important and depends on the optimization problem. There are no rules on it. Many researchers have dealt with this problematic and the proposed combinations of PSO parameters [14].

The set of parameters we have developed in these three models consists of the use of several variable parameters, which can be modified from the user interface dedicated for this, depending on the requirements of the optimization problem. A massive experiment was conducted to find the right set of parameters. The results are considered satisfactory. Below is the list of the PSO parameters used in our experiment for the three parallel models (see Tab. 3).

4 Results

The modification of the basic PSO algorithm for our three models concerns three essential points: the concept of the neighborhood (static, dynamic and evolutionary), the consistency of the parameters, and the parallel computation. These modifications of the algorithm improve its performance. Our algorithm is programmed in Java 1.8.

The graphs below show the results' detail of the average of 1000 executions: the values of execution time in seconds, the SR: the success rate which is the percentage of convergence of the function towards the right solution, and this concerns the basic PSO and the three parallel models on a set of ten functions. Empirical results are graphically illustrated in Figures 7 and 8.

4.1 Results' analysis

From the obtained results, one can observe that SPM algorithm performed much better than NPM and MPM algorithms in terms of the solution quality and parallel efficiency.

The SPM algorithm is developed to overcome the disadvantages of both NPM and MPM models by:

- Using the position generator (SOBOL) that allows a good distribution of the particles in the search space, resulting in a good exploration of the search space.
- Using a thread per neighborhood proved to be wise so as to avoid the unnecessary creation of threads at each iteration (which is costly in terms of computing time).
- Avoiding synchronization due to non-sharing of the global best at each iteration of the SPM algorithm.
- Using the concept of evolutionary neighborhood, which enables better exploitation of the search space.
- Providing the SPM algorithm with three stopping criteria allows for some optimization problems improving the solution quality and reducing the calculation time.

Table 3. Description of the used PSO parameters.

PSO parameters	NPM model	MPM model	SPM model
Swarm size	30	30	30
Number of iterations	50–80	50–80	50–80
Acceleration Constants	$C_1 = 1.25$, $C_2 = 2.25$, $C_3 = 2.25$	$C_1 = 1.25$, $C_2 = 2.25$, $C_3 = 0$	$C_1 = 1.25$, $C_2 = 0$, $C_3 = 2.25$
Communication topology	Ring	Fully connected	Ring
Inertia weight	Linearly decreasing $(0.4 - 0.2)$	Linearly decreasing $(0.4 - 0.2)$	Linearly decreasing $(0.4 - 0.2)$
Number of used threads	Depends on the objective function	Depends on the number of sub-spaces	Depends on the number of created neighborhoods
Stopping criteria	- The maximum number of iterations - The maximum number of iterations without improvement of Gbest	- The maximum number of iterations - The maximum number of iterations without improvement of Gbest	-A precision relative to the radius - A precision relative to the distance of Gbest - The maximum number of iterations without improvement of Gbest

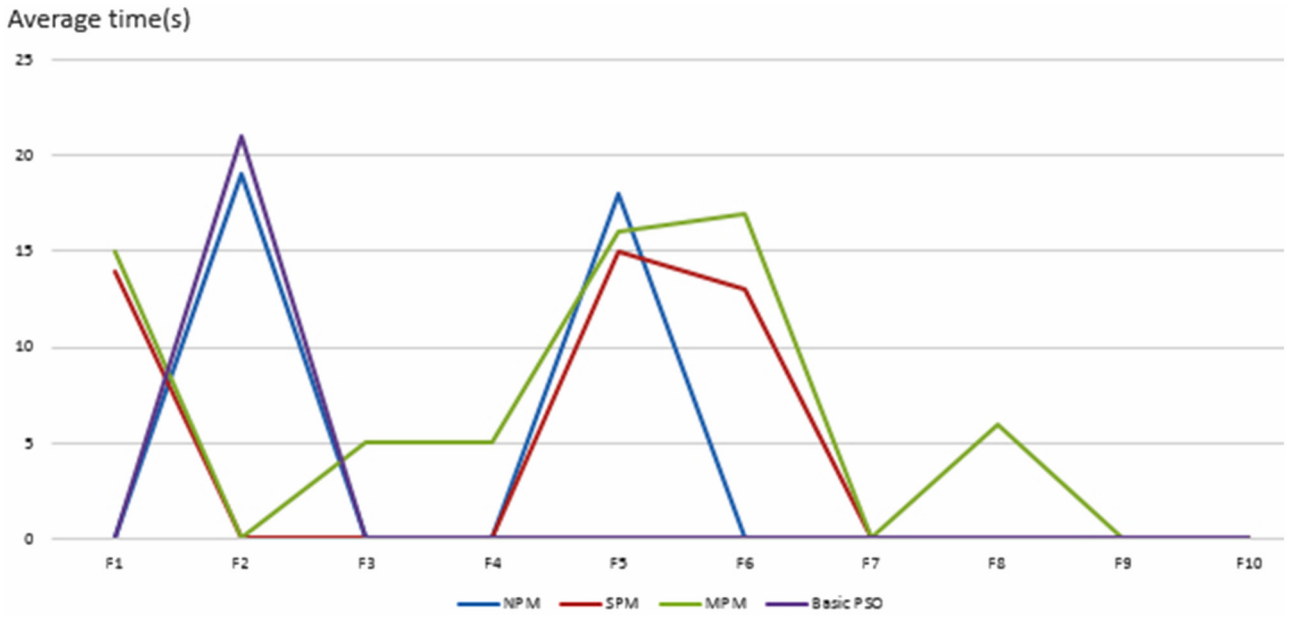

Fig. 7. Computation time performance for NPM, MPM, and SPM models.

The evaluation criteria such as the global performance, the solution quality, the robustness and the computation time of the three parallel models: NPM, MPM and SPM are compared. The conclusion is that the SPM algorithm has good optimization performance for all used test functions.

5 Conclusion

This paper deals with a description of a comparative study of three parallel models: NPM, MPM, and SPM based on the PSO algorithm. Although the PSO algorithm is very efficient, it has some disadvantages. The most studied ones are: premature convergence and high computing time.

The first NPM proposed parallel model uses a concept of dynamic neighborhood combined with parallel computing. The NPM model presented satisfactory results compared to the basic PSO; however, it has some disadvantages: synchronization cost and non-uniform distribution of particles in the search space.

To remedy these disadvantages, a second parallel model is proposed: MPM; it allows a better exploration and exploitation of the search space and presents satisfactory results compared to the NPM model. But the MPM model has some limitations, especially for complex optimization problems.

SPM is the third proposed model that combines two concepts: evolutionary neighborhood and parallel computing. This combination, according to our approach, has improved the results previously obtained from both NPM and MPM models.

Our three proposed models were tested on 10 benchmark functions. From the experimental results of these functions, it can be seen that our SPM algorithm

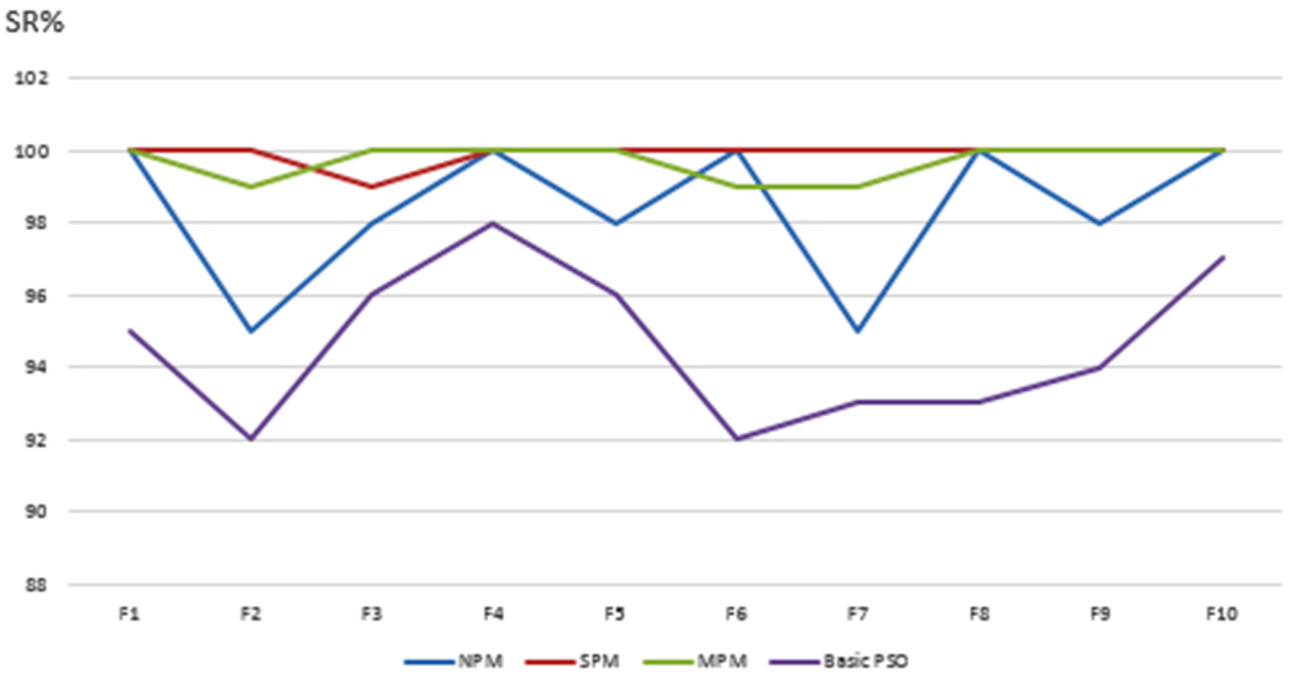

Fig. 8. Success performance curves for NPM, MPM and SPM models.

performed much better than NPM and MPM algorithms on the selected problems.

In future work, we intend to test the SPM model on real optimization problems.

References

1. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of the IEEE International Joint Conference on Neural Networks*, IEEE Press, vol. 8, 1995, pp. 1943–1948
2. J. Kennedy, R. Mendes, Population structure and particle swarm performance, in *Proceedings of the IEEE 2002 Congress on Evolutionary Computation*, 2002
3. T.-O. Ting, M.V.C. Rao, C.K. Loo, S.-S. Ngu, A new class of operators to accelerate particle swarm optimization, Proc. IEEE Congr. Evol. Comput. **4**, 2406–2410 (2003)
4. R. Poli, W.B. Langdon, O. Holland, Extending particle swarm optimization via genetic programming, in: M. Keijzer, A.G.B. Tettamanzi, P. Collet, J. van Hemert, M. Tomassini (Eds.), *Genetic Programming, LNCS* (Springer, Heidelberg, 2005), vol. 3447, pp. 291–300
5. T.R. Ramanan, M. Iqbal, K. Umarali. A particle swarm optimization approach for permutation flow shop scheduling problem, Int. J. Simul. Multisci. Des. Optim. **5**, A20 (2014)
6. N. Elhami, R. Ellaia, M. Itmi, Hybrid evolutionary optimization algorithm MPSO-SA, Int. J. Simul. Multisci. Des. Optim. **4**, 27–32 (2010)
7. M. Zemzami, N. Elhami, M. Itmi, N. Hmina, A new parallel approach for the exploitation of the search space based on PSO algorithm, in *International Colloquium in Information Science and Technology (CIST'16), Tangier, Morocco*, 2016
8. J. Chang, S. Chu, J. Roddick, J. Pan, A parallel particle swarm optimization algorithm with communication strategies, J. Inf. Sci. Eng. **21**, 809–818 (2005)
9. K. Byung-I, G. Alan, Parallel asynchronous particle swarm optimization, Int. J. Numer. Methods Eng. **67**, 578–595 (2006)
10. M. Rashid, A. Rauf Baig, PSOGP: A genetic programming based adaptable evolutionary hybrid particle swarm optimization, Int. J. Innov. Comput. Inf. Control **6**, 287–296 (2010)
11. P. Rabanal, I. Rodríguez, F. Rubio, Parallelizing particle swarm optimization in a functional programming environnement, Algorithms **7**, 554–581 (2014)
12. M. Zemzami, N. Elhami, M. Itmi, N. Hmina, Parallèlisation de la Méthode PSO: découpage de l'espace et traitement par lot des particules, in *International Workshop on New Services and Networks (WNSN'16), Khouribga, Morocco*, 2016
13. M.I. Aouad, Conception d'algorithmes hybrides pour l'optimisation de l'énergie mémoire dans les systèmes embarqués et de fonctions multimodales, Doctorat thesis, Henri Poincaré University- Nancy 1, France, 2011
14. M.E. Hyass, P. Hyass, Good parameters for particle swarm optimization, in: Laboratories Technical Report No. HL 1001, 2010

Cite this article as: Maria Zemzami, Norelislam El Hami, Mhamed Itmi, Nabil Hmina, A comparative study of three new parallel models based on the PSO algorithm, Int. J. Simul. Multidisci. Des. Optim. **11**, 5 (2020)