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/ijatcse05822019.pdf
https://doi.org/10.30534/ijatcse/2019/05822019

An evolutionary hybrid algorithm for complex optimization problems

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

This paper describes an evolutionary hybrid model linking two algorithms: Particle Swarm Optimization (PSO) and Simulated Annealing (SA). The basic idea behind using a hybrid model is improving the reliability of the obtained results from our first model, namely MPSO (Modified PSO) based on PSO algorithm, by adding SA algorithm which is quite popular for its powerful feature of effective escaping from the trap of local minima. MPSO model uses the concept of evolutionary neighborhoods associated to parallel computation, to overcome to the two essential disadvantages of PSO: high running time and premature convergence.

The presented algorithm has two essential operations: first running PSO algorithm in parallel using the new concept of evolutionary neighborhood to obtain a global best solution, then improving the results with SA algorithm to get the global optimal solution.

By testing this hybrid algorithm (H-MPSO-SA) on a set of standard benchmark functions and according to the obtained results, the program have given satisfactory results of the hybrid model compared to the basic PSO and MPSO algorithms.

Key words: Hybrid algorithm, metaheuristic, PSO, SA, Parallel computing.

1.INTRODUCTION

In recent decades, several optimization methods have emerged, often used to solve complex engineering problems in different fields [1]. PSO and SA are among the powerful metaheuristics used to solve real-world optimization problems, and have some forces and constraints, as well as for all other metaheuristics. Particle Swarm Optimization is a nature-inspired method based on the imitation of social interaction and creatures' communication such as bird flocks and fish schools [2]. It has shown distinguished performance for solving complex engineering problems such as structural and biomechanical optimizations, and it is now one of the most regularly used optimization algorithms. However, High running time and premature convergence are still great limitations for PSO, especially for complex optimization problems with a large search space and high dimension. In the basic PSO algorithm, particles are traveling in the search domain trying to find the most effective result, and updating their positions by their offspring no matter whether they are improved. In the search space, if a particle goes to an improved place, it can be changed by the updated. Yet if it goes to a bad place, it is still substituted by its offspring. Indeed, the most particles move to bad positions in most of the time, consequently the full swarm will diverge.

In this work, we propose an efficient hybrid algorithm, called H-MPSO -SA, to avoid premature convergence of PSO and to improve the MPSO model. H-MPSO-SA model is based on the idea that the PSO algorithm allows quick convergence, as long as SA technique brings the search away from local optima as a result of its forceful ability of local-search. Our executed model presents a good adaptation of PSO and SA parameters, parallel computation, the new concept of dynamic neighborhood and hybrid strategy. In our experimentations, the tests conducted on the program have given satisfactory results of the hybrid model compared to the basic PSO and MPSO models.

The remainder of this paper is organized as follows: In Section 2, we present the PSO and SA algorithms. In Section 3, we describe our hybrid approach based on PSO and SA algorithms. The testing and interpretation of results will be subject to Section 4, followed by a conclusion.

2.OVERVIEW OF PSO AND SA ALGORITHMS

2.1 Particle Swarm Optimization algorithm

PSO is a stochastic population-based method, impressed from the nature social behavior and dynamic activities with communication of animals to solve discrete and continuous optimization problems [2]. As delineated by its inventors: J. Kennedy (psychologist) and R. Eberhart (electrical engineer) in 1995 [2], "particle swarm algorithm imitates human (or insects) social behavior. Individuals interact with one another while learning from their own experience, and gradually the population members move into better regions of the problem space".

A. Formalization

PSO algorithm uses a set of agents (named particles) that represent a swarm traveling across the multidimensional search space in the search of the optimum. A particle is a point, and it is represented by its current position Xa = (xa1, xa2,…,xab), and its current velocity Va = (va1 ,va2 ,…,vab) a =1,2,…,s; b=1,2,…,m where s is the swarm size.

For each iteration, PSO makes a search of the optimum by firstly moving the particles, then evaluating the new position fitness.
All the particles search for better positions by moving around the search space, changing their velocities and positions with every iteration.
The movement of the particle by basic PSO depends on three important terms: the particle's previous velocity, the particle's individual best position and the global best position.
Updating positions and velocities respectively by formulas (1) and (2):

V(k+1) = V(k) + C1r1(Pbest(k) - X(k)) + C2r2(Gbest(k) - X(k)) (1)

X(k+1) = X(k) + V(k+1) (2)

With:
D represents the dimension of the optimization problem,
XaD(k) represents the particle's position at time step k,
VaD(k) represents the particle's velocity at time step k,
PbestD(k) represents the best memorized personal particle position, at time step k,
GbestD(k) represents the best memorized swarm position, at time step k,
C1, C2 and C3 are acceleration factors,
r1 and r2 are random numbers drawn from the interval [0,1],

PSO algorithm has been at the center of interest of many researchers in the field of optimization. Several improvements have been proposed others are still possible. While in its current state, PSO algorithm remains a very powerful method. In an environment based on the PSO method, particles modify their velocities and positions dynamically by moving in a N-dimensional domain space until the stopping criterion is reached (figure 1). (For each particle, taking into consideration its movement information and those of its neighborhoods) [3].

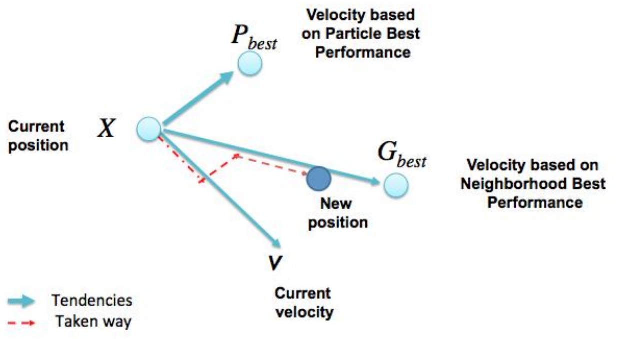

Velocity based on Particle Best Performance
P_{best}
Current position
X
G_{best}
Velocity based on Neighborhood Best Performance
New position
V
Current velocity
Tendencies
Taken way

Figure 1: The particle's movement by classical PSO

B. *Algorithm*

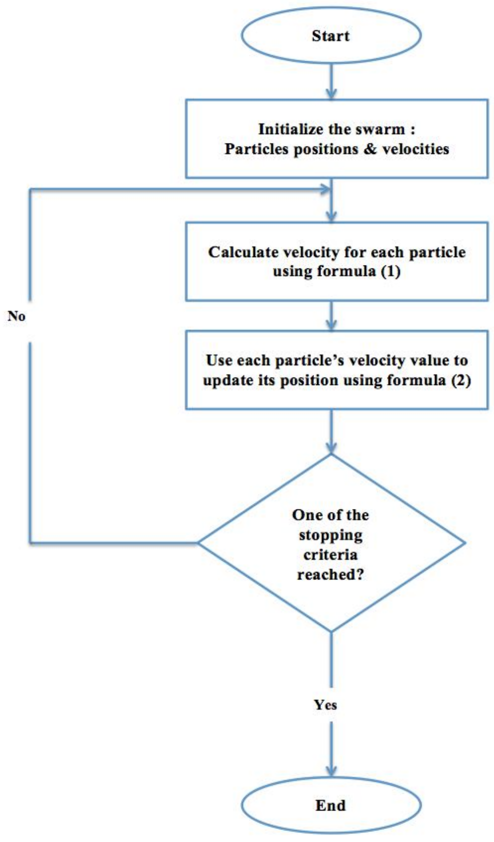

Start
Initialize the swarm : Particles positions & velocities
Calculate velocity for each particle using formula (1)
Use each particle's velocity value to update its position using formula (2)
One of the stopping criteria reached?
No
Yes
End

Figure 2: The PSO algorithm flowchart

The classical PSO algorithm is a randomized process invented by [2] (figure 2). The summary of the main steps of the algorithm is as follows:

(1) Creating the Swarm: Initializing particle's velocities and positions. The particles' positions must be stochastically distributed in the search domain.
(2) Calculating the particle's velocity using Formula 1.
(3) Updating the particle's position using Formula 2.
(4) Repeat steps 2 and 3 until convergence.

C. *The concept of neighborhood*

The neighborhood of a particle is the subset of particles of the swarm with which it has direct communication. This network of relationships between all particles is known as sociometry, or the topology of the swarm.
We can distinguish two main types of neighborhoods:
The social and static neighborhood: as its name suggests, it represents the social adjacency. Neighborhoods are no longer the distance expression but the expression of the exchange of information, the neighbors are defined in the beginning of the algorithm and are not updated by the following.
It should be noted that this is an easy type of neighbourhood to implement.
The geographical and dynamic neighborhood: as its name suggests, it represents the geographical adjacency. At each iteration, neighbors must be updated from a preset distance in the search domain. This is the kind of neighborhood that was developed in our model.

In this case, the change of velocity in the equation (1) is changed by introducing a new term in the formula. Proposed by [4], its illustration appears in Figure. 3 (see [5]).

V(k+1) = V(k) + C1r1(Pbest(t) - X(k)) + C2r2(Gbest(k) - X(k))+ C3r3(Pnbest(k) - X(k)) (3)

With: Pnbest: the neighborhood best position; C3: the acceleration coefficient, r3: random number between 0 and 1.

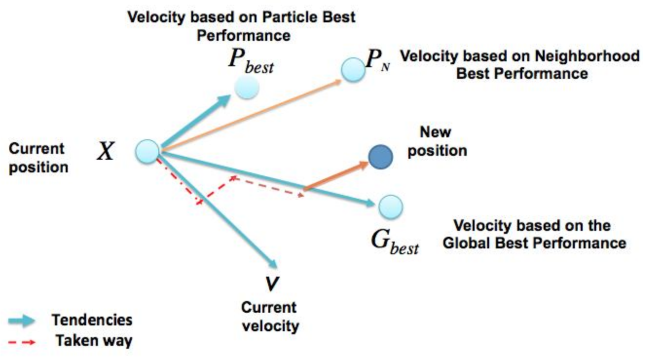

Velocity based on Particle Best Performance
P_{best}
P_N
Velocity based on Neighborhood Best Performance
New position
Current position
X
Velocity based on the Global Best Performance
G_{best}
V
Current velocity
Tendencies
Taken way

Figure 3: The particle's movement by PSO Modified

2.2 Simulated Annealing algorithm

SA is a probabilistic variant to approximate global optimization of a given function in a large search space. It is developed in 1983 by [6] to solve large-scale combinatorial optimization problems. It is motivated by an analogy to annealing in solids, a technique in which a material (usually a metal) is subjected to a controlled heating and a slow cooling process to improve the material inner structure and consequently to increase its toughness.

The principle of SA method is as follows: first we start by heating the metal to a certain temperature where it becomes liquid (the atoms can therefore circulate freely). After reaching this stage, the temperature is lowered very slowly to obtain a solid. If this drop in temperature is abrupt, then glass is obtained "amorphous solid state"; if it is the opposite case, this drop in temperature is very slow (leaving the atoms time to reach statistical equilibrium), we will obtain "crystalline solid state": structures more and more regular, until reaching a state of minimum energy corresponding to the perfect structure of a crystal, it is said that the system is "frozen". In fact, the thermo dynamicists have noticed that a sudden drop in the temperature of a liquid causes a reproduction of a local optimum, i.e. an amorphous structure. While a gradual decrease in the temperature of the liquid leads to a global optimum, i.e. a well constructed structure.

This is the idea taken into consideration by the metallurgists who know that if the metal cools too quickly, it will contain many microscopic defects and if it cools slowly they will get a well-ordered structure. This method is implemented in optimization to find the local extrema of a function.

A. Algorithm

SA algorithm is based on Metropolis algorithm, it starts with an initial value of temperature and a random initial solution, and then we start looping until our stopping criterion is reached. In general either the system has adequately cooled, or a good-enough solution has been found. Hence, we select a neighbor bringing a small modification to our current solution. After that, we decide whether to move to that neighbor solution using the formula (4) "acceptance probability". Finally, we decrease the temperature and continue looping.

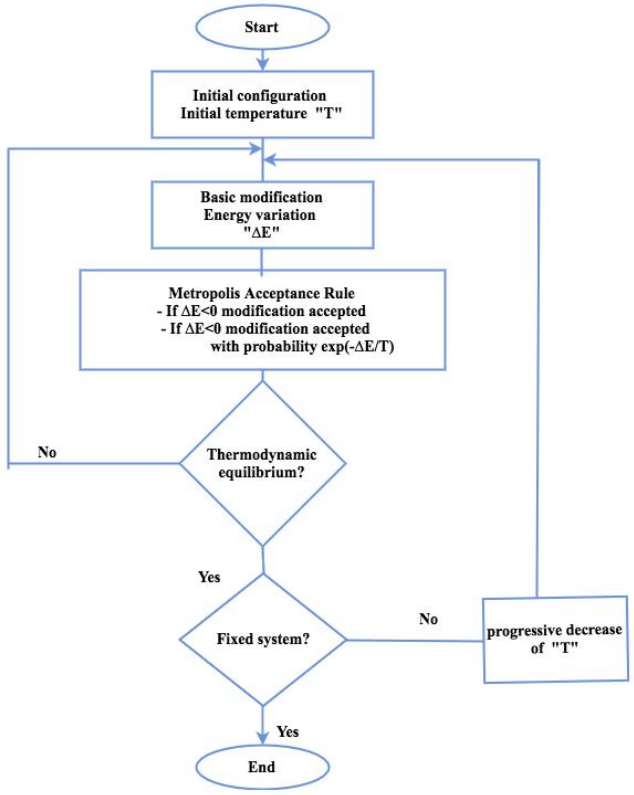

Start
Initial configuration
Initial temperature "T"
Basic modification
Energy variation
"ΔE"
Metropolis Acceptance Rule
- If ΔE<0 modification accepted
- If ΔE<0 modification accepted with probability exp(-ΔE/T)
Thermodynamic equilibrium?
No
Yes
Fixed system?
No
progressive decrease of "T"
Yes
End

Figure 4: Simulated Annealing algorithm flowchart

The acceptance probability of an unimproved movement is:

$$P(\Delta E,T)=e^{-\Delta E/T}>R \quad (4)$$

Where:
E is the change in the optimization function,
T represents the value of current temperature, and
R represents a uniform random number drawn from the interval [0,1].
The SA algorithm can be described by Figure 4 [7].

3.THE PROPOSED HYBRID APPROACH

This section presents a new hybrid H-MPSO-SA algorithm based on using the Particle Swarm Optimization techniques in conjunction with the Simulated Annealing approach. In fact, by using PSO with SA, the advantages of both SA algorithm (for its strong local-search ability) and PSO algorithm (for its strong global-search ability) are combined. This is the basic idea of the H-MPSO-SA (Figure 6).

In our algorithm, there are two main parts:

The first one called MPSO relates to the processing based on PSO algorithm using the concept of evolutionary neighborhoods associated to parallel computation.

The second part is about the processing of SA algorithm used in order to avoid being trapped into local minimum and to increase the diversity of particles.

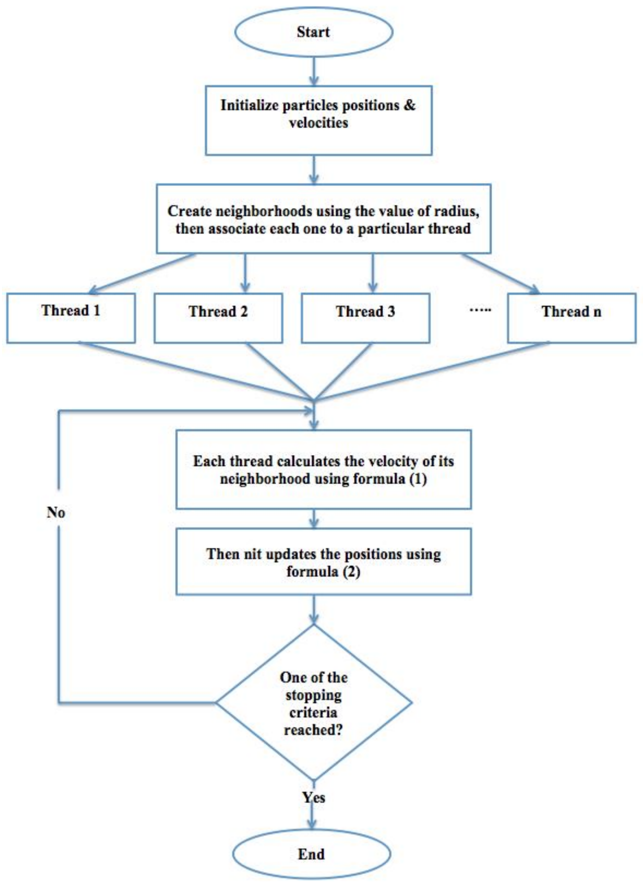

Start
Initialize particles positions & velocities
Create neighborhoods using the value of radius, then associate each one to a particular thread
Thread 1
Thread 2
Thread 3
.....
Thread n
Each thread calculates the velocity of its neighborhood using formula (1)
Then nit updates the positions using formula (2)
No
One of the stopping criteria reached?
Yes
End

Figure 5: Modified PSO model flowchart

The performance of the basic PSO algorithm is affected, (all calculations are done sequentially) especially for complex optimization problems. From here comes the idea of parallelization for PSO to improve its performance.

Several researchers in the field of metaheuristics have been interested in the concept of parallelization, different parallel PSO models have been proposed [8]-[12], for our parallel implementation allowing the parallelization of calculations. Indeed, threads, kind of processes perform calculations in parallel on sets of particles situated in several neighborhoods. Each thread executes the processing of an iteration for its particles' group, and then waits for the other threads to finish their processing for the purpose of updating the neighborhoods and starting a new iteration. This scenario is repeated until a sufficient solution is obtained, "the stopping criterion is reached".

For our parallel approach, the neighborhoods have the shape of spheres, which are updated at each iteration of the algorithm: their centers evolve and the value of radius changes depending on conditions related to the radius value, the search space and the number of neighborhoods.

Figure 5 is a flowchart of the MPSO model. The reader is referred to [13] for more details about this approach.

To test the proposed MPSO model, we studied the problem of the electricity transport, in particular, the electric pylon example, then we have applied our approach based on PSO algorithm for this example. According to the experiments related to this optimization problem, MPSO algorithm is powerful and surpasses classical PSO in terms of convergence speed, constraint handling, precision, and solution quality [14] [15].

To escape the stagnation of the MPSO algorithm in local optima, we propose an efficient hybrid model, called H-MPSO-SA, based on the idea that PSO algorithm assured fast convergence, while SA algorithm brings the search out of local optima because of its strong local-search ability. Our H-MPSO-SA algorithm does not use the SA algorithm in along all PSO algorithm iterations, but the SA algorithm performs its processing after a specific number of iterations (predefined before starting the program), and then PSO algorithm continue its calculations. We found that it is more judicious than using it at each PSO iteration.

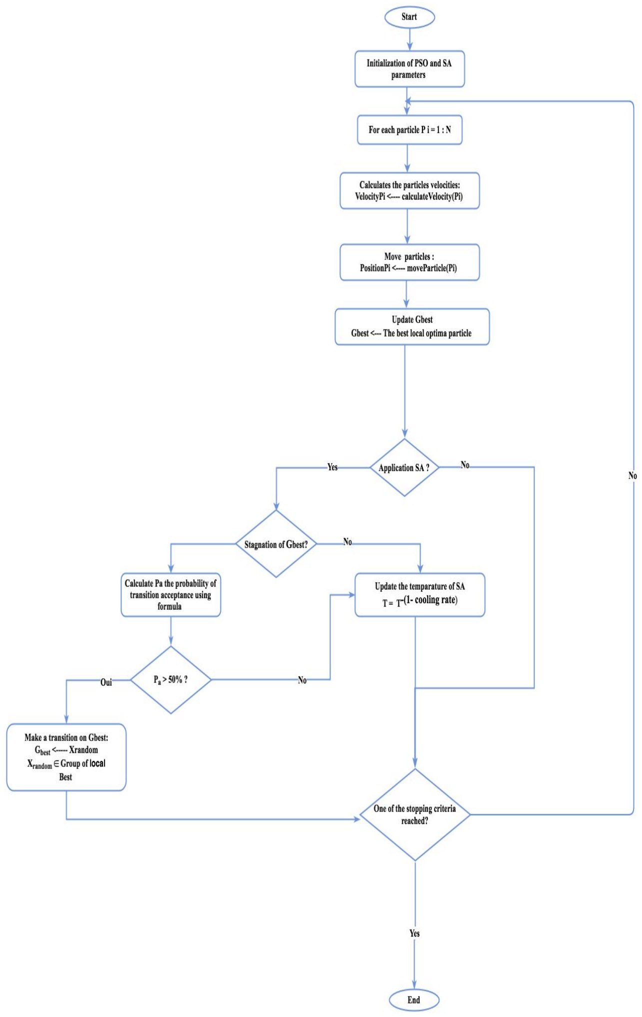

Start
Initialization of PSO and SA parameters
For each particle P i = 1 : N
Calculates the particles velocities: VelocityPi <---- calculateVelocity(Pi)
Move particles : PositionPi <---- moveParticle(Pi)
Update Gbest Gbest <--- The best local optima particle
Application SA ?
Yes
No
No
Stagnation of Gbest?
No
Calculate Pa the probability of transition acceptance using formula
Update the temparature of SA T= T-(1- cooling rate)
P_a > 50% ?
Oui
No
Make a transition on Gbest: G_best <----- Xrandom X_random ∈ Group of local Best
One of the stopping criteria reached?
Yes
End

Figure 6: Hybrid MPSOSA flowchart

If in each iteration we'd used both PSO and SA, then SA will try to diversify the points and PSO will try to converge the points at the same time, which will in turn delay the convergence of PSO as well as the capabilities of SA will also not be effective. So, we have applied PSO at the first, and applied SA after every 'X' PSO iterations, in order to give SA the possibility to escape from the local optima and diversify the prematurely converged particles in the search space.

Therefore, the hybrid model is designed to solve complex optimization problems, with several local optima and a large search domain.
On the one hand thanks to parallel PSO, our model is able to ensure fast convergence (most of the time), on the other hand, the use of SA algorithm allow escaping from a local optimum.
For our model, SA algorithm is applied to the last best solution found so far, each 'X' iterations that is predefined at the beginning of the program depending on the optimization problem; in order to overcome the PSO premature convergence.
The particularity of the hybridization presented in this model is based on the combination of the advantages of PSO and SA algorithms. Choosing good PSO parameters, particularly the notion of geographical evolutionary neighborhood, which allows good exploration and exploitation of the search space (by well sharing information between different neighborhoods) especially with the presence of the SA algorithm known by its large local search capacity. Parallel computation is used to improve the quality of the solution found in terms of computing time.

The flowchart of the suggested model H-MPSO-SA is presented in Figure 6. The reader is referred to [16]-[20] for other versions of hybrid PSO and SA algorithms.
(1): « T » the initial temperature of the system (it decreases after each iteration to stabilize the system).
(2): « X » the application frequency of the SA algorithm, so we can optimize the calculation time.
(3): V(k+1) = V(k) + C1r1(Pbest(k) - X(k)) + C2r2(Gbest(k)-X(k))+ C3r3(Pn(k) - X(k))
(4): X(k+1) = X(k) + V(k+1)
(5): The probability of acceptance = Exp(((difference-energy) / (temperature))) = $e^{-\Delta E/T}$
With:
Exp: the exponential function.
Difference-energy = energy of Global best – energy of random Local best.
Temperature: Current temperature of SA algorithm.

4.EXPERIMENTAL RESULTS

For our model based on PSO and SA algorithms, the modification includes four categories: a new version of evolutionary neighborhood, the concept of parallelization, adjustment of the PSO and SA parameters, and the hybrid model. These modifications of H-MPSO-SA algorithm improve its performance.

4.1. Benchmark problems

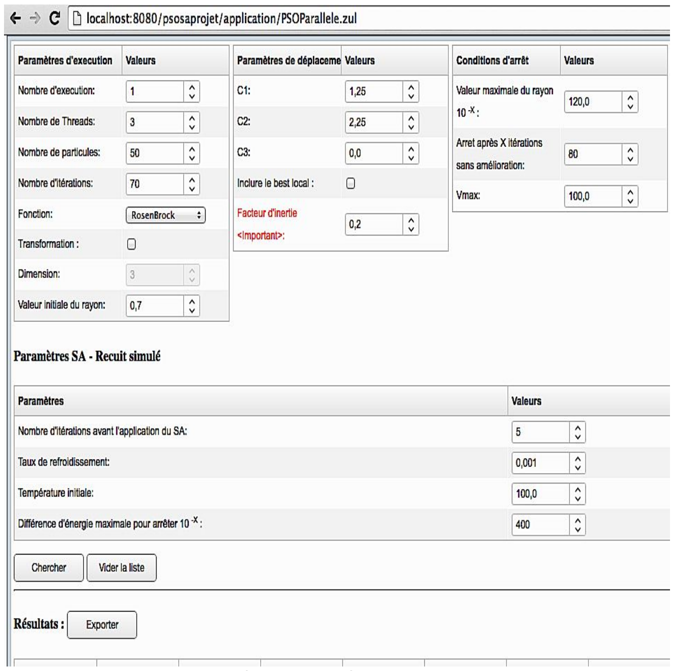
In literature, there are collections of problems specifically used to test the performance of the optimization methods, such as: accuracy, time consuming, the solution quality, precision, etc.
Ten test functions (f1-f10) are used in this paper to prove the performance of our hybrid model.
These set of functions were applied on different optimization problems and provides a reliable source of credible data that can be used for the purpose of optimization techniques.
For each of these test functions, there could be many local optima as well as one or more global optima in their solution space. As we keep increasing the number of dimensions, the problem becomes more complex, more local optima are likely to occur and it leads to delay in converging to the correct global solution 'the optimum' for that function. For this study 2, 5, 10, 15, 20 and 30 dimensional functions are taken.

Table 1: Description of the used functions in our experiments

Function	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	±π	-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

Like all Evolutionary Algorithms, PSO and SA methods have a set of parameters, which is to be defined by the user. Our H-MPSO-SA program has an Interface Human Machine where the user can choose all the PSO and SA parameters according to the user need, such as: the definition of the global search space, the dimension of the optimization problem, the number of used particles, the optimization problem (i.e. the objective function), the communication topology, initial SA temperature, acceleration coefficients, the cooling rate, the stopping criterion etc. (see Figure 7).
These parameters vary according to the complexity of the optimization problem.



Paramètres d'execution	Valeurs
Nombre d'execution:	1
Nombre de Threads:	3
Nombre de particules:	50
Nombre d'itérations:	70
Fonction:	RosenBrock
Transformation :	☐
Dimension:	3
Valeur initiale du rayon:	0,7

Paramètres de déplaceme	Valeurs
C1:	1,25
C2:	2,25
C3:	0,0
Inclure le best local :	☐
Facteur d'inertie <Important>:	0,2

Conditions d'arrêt	Valeurs
Valeur maximale du rayon 10^{-X}:	120,0
Arret après X itérations sans amélioration:	80
Vmax:	100,0

Paramètres SA - Recuit simulé

Paramètres	Valeurs
Nombre d'itérations avant l'application du SA:	5
Taux de refroidissement:	0,001
Température initiale:	100,0
Différence d'énergie maximale pour arrêter 10^{-X}:	400

Résultats :

Figure 7: Screenshot of user interface (PSO and SA parameters)

For this study, a fixed population size of forty particles is taken for all the optimization problems, which gave reasonably good results. Similarly various examples are available on the variations done in inertia weight factor and acceleration constants coefficients.

For the present study, the list of the used PSO and SA parameters, which gave during the experiments, sufficiently good results are mentioned below:

- Number of particles: 40.
- Number of iterations: 1500.
- Communication topology: Ring.
- Acceleration factors: c1 = 1.25, c2 = 2.25, c3: 2.25.
- Initial SA temperature: 100
- Cooling rate: 0.001.
- X=5 (With X: the number of PSO iterations before starting the SA processing).

-Number of threads: Depends on the number of created neighborhoods. (Each thread is assigned a neighborhood processing).

-Radius value: Depends on the objective function search space.

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).

4.3. Results

The presented results are related to a set of 10 test functions, each function contains a number of local and global minima. For this experiment we used an average of 100000 function evaluations.

Programming language used is JAVA. 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 reducing computation time and a better use of material resources of the machine.

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.

In the following graphs we give the detail of the average of results namely the values of the execution time in seconds (Fig. 8), the SR (Success Rate): the success rate represents the percentage of function convergence to the right solution (Fig. 9), and (Fig. 10) SD (Standard Deviation) represents the standard deviation computed using the formula:

$$SD = \sqrt{1/n(\sum_{i=1}^{n}(Xi - X^*)^2)}$$

Where X* is the optimal solution and Xi the solution found for each test for the basic PSO, MPSO and hybrid PSO-SA model on a set of ten functions.

According to the results, we can say that the proposed H-MPSO-SA algorithm provides the optimal solution with a higher probability and the computation time in H-MPSO-SA is lower than in the basic PSO and MPSO. The graphical results are illustrated in the three following figures.

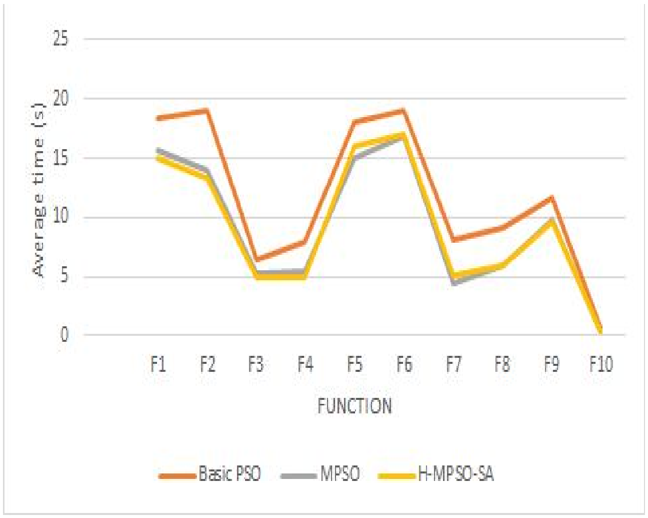


Figure 8: Computation time performance for PSO, MPSO and H-MPSO-SA

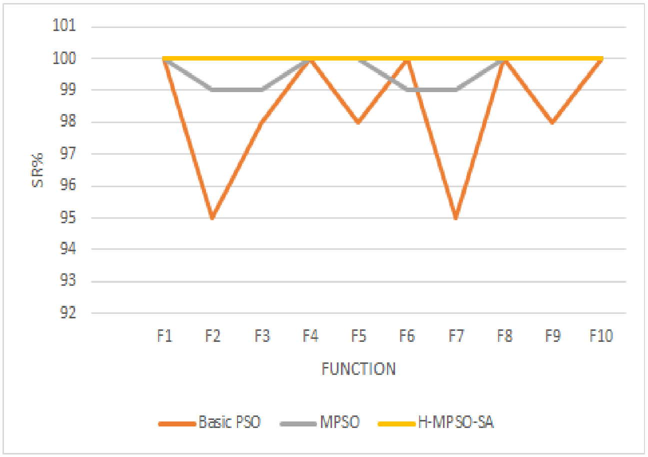


Figure 9: Success performance curves for PSO, MPSO and H-MPSO-SA

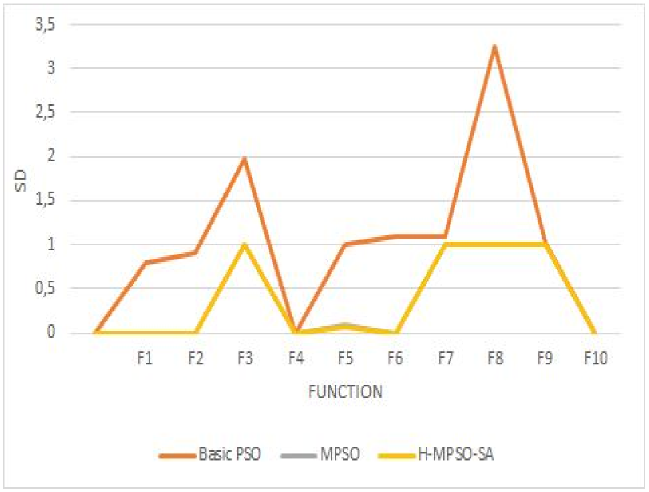


Figure 10: Standard deviation study for PSO, MPSO and H-MPSO-SA

5. CONCLUSIONS

For this work, an efficient hybrid version of Particle Swarm Optimization and Simulated Annealing algorithms is presented. In this model, for the PSO part, two new concepts are linked: dynamic neighborhoods and parallel computation, in order to improve the PSO performance and to avoid its weaknesses (high running time and premature convergence). PSO is a robust metaheuristic based on population solutions. It is used in several fields and proved high performance.

On the other hand, SA algorithm is very known for its powerful local-search capacity, and it is used to ensures that the search jump out of local optima.

This hybrid approach makes full use of the local and global search optimization ability of both SA and PSO respectively and overcomes the limitations of each algorithm separately possesses. Through application of SA to PSO, H-MPSO-SA model is capable of escaping from local optima and succeeds in converging into the global optima in the search space in a very good time consuming.

The proposed model was tested on a set of optimization functions. From the obtained results, it can conclude that H-MPSO-SA model performed much better than a classical PSO and MPSO on this series of optimization problems.

Finally, in the future we intend to compare H-MPSO-SA algorithm with other hybrid algorithms and to test the program on 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. Fatiha Djaafar, Baghdad Hadri, Ghalem Bachir. **Optimization and Comparison between the Efficiency of InGaP and GaAs Single Solar Cells with and without a Window Layer.** International Journal of Advanced Trends in Computer Science and Engineering, vol. 7, no. 4, pp.61- 66. 2018. https://doi.org/10.30534/ijatcse/2018/01742018
2. 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.
3. Y. Cooren. **Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique.** Doctorat thesis, University of Paris 12 Val de Marne, France, 2008.
4. B. Bochnek et P. Fory's. **Structural optimization for post buckling behavior using particle swarms.** Struct Multidisc Optim. Pages 521-531, 2006. https://doi.org/10.1007/s00158-006-0044-8
5. N. Elhami, R. Ellaia, M. Itmi. **Hybrid Evolutionary Optimization Algorithm MPSO-SA.** In: International Journal of Simulation and Multidisciplinary Optimimisation, Vol. 4, pp. 27- 32, 2010. https://doi.org/10.1051/ijsmdo/2010004
6. S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. **Optimization by simulated annealing.** Science, vol. 220, no. 4598, pp. 671–680, 1983. https://doi.org/10.1126/science.220.4598.671
7. S. Zhan, J. Lin, Z. Zhang, and Y. Zhong. **List-Based Simulated Annealing Algorithm for Traveling Salesman Problem.** In Hindawi Publishing Corporation Computational Intelligence and Neuroscience Volume 2016, Article ID 1712630. 2016. https://doi.org/10.1155/2016/1712630
8. 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
9. 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
10. 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.
11. M.Zemzami, N.Elhami, M.Itmi and 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.
12. 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
13. M.Zemzami, N.Elhami, M.Itmi and N.Hmina. **Parallelization of the PSO algorithm on evolutionary neighborhoods.** In: *International Conference on*

Modeling, Optimization and Simulation (MOSIM'16). Montréal. Canada. 2016.

14. 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.** In *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
15. 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
16. G. Yang, D. Chen, and G. Zhou. **A new hybrid algorithm of particle swarm optimization.** In Lecture Notes in Computer Science, vol. 4115, pp. 50–60, 2006. https://doi.org/10.1007/11816102_6
17. M. Bahrepour, E. Mahdipour, R. Cheloi, and M. Yaghoobi. **Super-sapso: a new sa-based pso algorithm.** in Applications of Soft Computing, vol. 58, pp. 423–430, 2009. https://doi.org/10.1007/978-3-540-89619-7_41
18. W. J. Xia and Z. M. Wu. **A hybrid particle swarm optimization approach for the job-shop scheduling problem.** International Journal of Advanced Manufacturing Technology, vol. 29, no. 3- 4, pp. 360–366, 2006. https://doi.org/10.1007/s00170-005-2513-4
19. X. Wang and J. Li. **Hybrid particle swarm optimization with simulated annealing.** In Proceedings of the *3rd International Conference on Machine Learning and Cybernetics (ICMLC '04),* vol. 4, pp. 2402–2405, 2004.
20. L. Idoumghar, M. Melkemi, R. Schott, M.I Aouad. **Hybrid PSO-SA Type Algorithms for Multimodal Function Optimization and Reducing Energy Consumption in Embedded Systems**. Applied Computational Intelligence and Soft Computing, 2011, 2011, pp.Article ID 138078. https://doi.org/10.1155/2011/138078