Metropolis Particle Swarm Optimization Algorithm with Mutation Operator For Global Optimization Problems

L. IDOUMGHAR
LMIA-MAGE
INRIA-Grand Est
Université de Haute Alsace
Mulhouse, France
lhassane.idoumghar@uha.fr

M. IDRISSI AOUAD
INRIA Nancy - LORIA.
Université Henri Poincaré
Villers-Lès-Nancy, France
Maha.IdrissiAouad@inria.fr

M. MELKEMI
LMIA-MAGE
Université de Haute Alsace
Mulhouse, France
mahmoud.melkemi@uha.fr

R. SCHOTT
IECN and LORIA
Université Henri Poincaré
Vandoeuvre-lès- Nancy, France
rene.schott@loria.fr

Abstract

When a local optimal solution is reached with classical Particle Swarm Optimization (PSO), all particles in the swarm gather around it, and escaping from this local optima becomes difficult. To avoid premature convergence of PSO, we present in this paper a novel variant of PSO algorithm, called MPSOM, that uses Metropolis equation to update local best solutions (lbest) of each particle and uses Mutation operator to escape from local optima. The proposed MPSOM algorithm is validated on seven standard benchmark functions and used to solve the problem of reducing memory energy consumption in embedded systems. The numerical results show that our approach outperforms several recently published algorithms.

1. Introduction

Global optimization[1] (GO) is the branch of applied mathematics and numerical analysis that focuses on, well, optimization. GO can be defined as follows: $Minimize f(x) : S \rightarrow R$ where $f(.)$ is the objective function (called also fitness value) which is subject to optimization, $S \subset \Re^D$ and D is the dimension of the search space S. Solving global optimization problem means that we need to find $x^* \in S$ such that $f(x^*) \leq f(x), \forall x \in S$. x^* is called the global minimizer of $f(.)$ and $f(x^*)$ is called the global minimum value of $f(.)$. Finding this global minimum is very difficult dues to the existence of several local optima. Over the last decades, several optimization heuristics have been proposed for solving GO. Among the many heuristics let us mention Simulated Annealing (SA) [2], Genetic Algorithm [21] and optimization algorithms that make use of social or evolutionary behaviors like Particle Swarm Optimization (PSO) [19, 22, 1]. PSO is quite popular heuristics for solving complex optimization problems but this method has strengths and limitations principally premature convergence. To avoid premature convergence [23] of PSO, many works in the PSO community try to hybridize PSO with other heuristic algorithm [19, 22, 18]. Particle Swarm Optimization (PSO) is based on the social behavior of individuals living together in groups. Each individual tries to improve itself by observing other group members and imitating the better ones. That way, the group members are performing an optimization procedure which is described in [3]. The performance of the algorithm depends on the way the particles (i.e., potential solutions to an optimization problem) move in the search space with a velocity that is updated iteratively. Large body of research in the field has been devoted to the analysis and proposal of different motion rules (see [19, 4, 5, 6] for recent accounts of PSO research).

In this paper, we present a new algorithm, called MPSOM, that makes full use of the exploration ability of PSO and incorporates a mutation operator and Metropolis rule to jump out from local optimum. MPSOM has been validated on 7 benchmark functions [7] and compared with PSO al-

[1] http://www.it-weise.de/projects/book.pdf

gorithms variants described in [19, 18, 24]. We also use MPSOM algorithm to solve the problem of reducing memory energy consumption in embedded systems. The simulation results show that MPSOM outperforms the above mentioned algorithms. Hence MPSOM is a good alternative for dealing with complex numerical function optimization problems. This paper is organized as follows. Section 2 introduces briefly PSO algorithm. Section 3 is devoted to a precise detailed description of MPSOM . In Section 4, a series of numerical experiments regarding solution quality, convergence rate and robustness are conducted to show the superiority of our MPSOM algorithm. Discussion and results obtained about solving the problem of reducing Memory energy consumption in embedded systems are presented in Section 5. Finally, conclusions and further research aspects are given in Section 6.

2. Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [10], inspired by social behavior patterns of organisms that live and interact within large groups. In particular, it incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior.

The idea of PSO algorithm is that particles move through the search space with velocities which are dynamically adjusted according to their historical behaviors. Therefore, the particles have the tendency to move towards the better and better search area over the course of search process. PSO algorithm starts with a group of random (or not) particles (solutions) and then searches for optima by updating each generation. Each particle is treated as a volume-less particle (a point) in the n- dimensional search space. The i^{th} particle is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{in})$. At each generation, each particle is updated by following two $best$ values:

- The first one is the best solution (fitness) it has achieved so far (The fitness value is also stored). This value is called $cbest$.
- Another best value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called $gbest$. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called $lbest$.

At each iteration, theses two best values are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new move for the particle. The portion of the adjustment to the velocity influenced by the individual's previous best position ($cbest$) is considered the *cognition* component, and the portion influenced by the best in the neighborhood ($lbest$ or $gbest$) is the social component.

With the addition of the inertia weight factor, ω, by Shi and Eberhart [11] (for balancing the global and the local search), these equations are:

$$v_{ij} = \omega \times v_{ij} + c_1 \times rand \times (cbest_{ij} - x_{ij}) + c_2 \times rand \times (gbest_{ij} - x_{ij}) \quad (1)$$

$$x_{ij} = x_{ij} + v_{ij} \quad (2)$$

where $rand$ is a random number independently generated within the range of [0,1] and c_1 and c_2 are two learning factors which control the influence of the social and cognitive components (usually, c_1 = c_2 = 2).

In Equation 1 if the sum on the right side exceeds some fixed $Vmax$ value, then the velocity on that dimension is assigned to be $\pm V_imax$. Thus, particles' velocities are clamped to the range of $[-V_imax; V_imax]$ which serves as a constraint to control the global exploration ability of PSO algorithm. This also reduces the likelihood of particles for leaving the search space. Note that this does not restrict the values of x_i to the range $[-V_imax; V_imax]$; it only limits the maximum distance that a particle will move during one iteration.

3. MPSOM Algorithm

This section presents MPSOM algorithm that makes full use of the exploration ability of PSO and incorporates a mutation operator and Metropolis rule to jump out from local optimum. The mutation operator is introduced to PSO every K iterations if no improvement of the global best solution does occur. The value of K is predefined to 60 according to our experimentations.

The principle of MPSO algorithm is works as illustrated in Algorithm 1, where:

Description of a particle: Each particle (solution) is represented by its:

- current position $X \in S$ represented by $n > 0$ components, i.e., $X = (x_1, x_2, , x_n)$, where $i = 1, 2, ..., n$ and n represents the dimension of the optimization problem to solve.
- previous best solution founded so far (called $cbest$).
- velocity that corresponds to the rate of the position change.

Initial Swarm: Initial Swarm corresponds to population of particles that will evolve. Each particle x_i is initialized with uniform random value between the lower

```
Initialize swarm_size particles
Evaluate(Swarm)
stop_criterion←maximum evaluation functions
noImprove ← 0,
Initialize inertia factor w ← w_0
Initialize Initial Temperature T ← T_0
while Not stop_criterion do
    Sort(Swarm)
    if nonimprove < k then
        for each particle i ← 1 to swarm_size do
            Accept(X, cbest, T)
            Update velocity according equation (4)
            Enforce velocity bounds
            Update particle position according equation (2)
            Enforce position bounds
        end
    else
        // Mutation operator
        noImprove ← 0
        for each particle i ← 1 to swarm_size do
            Initialize its velocity to maximum velocity
            allowed
        end
    end
    Evaluate(Swarm)
    if there is no improvement of global best solution then
        noImprove ← noImprove + 1
    else
        Update global best solution
        noImprove ← 0
    end
    Update inertia factor w by using equation 6
    Update (T)
    Update (stop_criterion)
end
```

Algorithm 1: MPSOM Algorithm.

and upper boundaries of the interval defining the optimization problem.

Evaluate function: Evaluate (or fitness) function in MSOSM algorithm is typically the objective function that we want to minimize in the problem.

- **Sort**: all particles of the swarm are sorted in decreasing order of their objective function.

  Accept function: Function $Accept(cbest, X, T)$ is decided by the acceptance probability given by equation 3, which is the probability of accepting current position of one particle as its $cbest$.

  The probability of accepting a new solution is given as follows:

$$p = \begin{cases} 1 & f(X) \leq f(cbest) \text{ or} \\ & \quad rand \times (1 + e^{\frac{f(X)-f(cbest)}{T}}) < 2.0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

  where:

  - T is the temperature. It plays a similar role as the temperature in the physical annealing process[12, 2]
  - $rand$ is a random number independently generated within the range of $[0, 1]$.

- update velocity: we propose in this work topological neighborhood which is determined dynamically during the search process, according to the objective function of the swarm. After we had sorted the Swarm population in decreasing order of their objective function, for each i^{th} particle, a set N_i of its neighbors is defined as: $N_i = \{particle\ k\ /\ k \geq i\ and\ k \leq swarm\_size\}$. Then, to adjust velocity of i^th particle, we used the following equation:

$$v_{ij} = \omega \times v_{ij} + c_1 \times rand \times (cbest_{ij} - x_{ij}) + min(V_{max}, \sum_{k=i}^{swarm\_size} \frac{cbest_{kj} - x_{ij}}{k}) \quad (4)$$

  In Equation 4, the social component of i^{th} particle is computed as a weighted average of all best particles in N_i. In so doing, a particle is not only influenced by the global best solution. The particle adjusts its speed according to the weighetd average of all solutions which are better than its ones. This means simply that it's better for particle to follow a group of particles rather than a signle one (this ins of course common sens).

  Dire que cette moyenne permet d'viter une particule d'tre influencer uniquement par la meilleure

solution. Elle ajuste donc sa vitesse en fonction de la moyenne pondre de toutes les solution qui sont meilleure qu'elle. Intuitivement, il vaut mieux suivre la tendance d'un groupe que de suivre 1 un seul individu!

- Temperature update: to avoid our algorithm getting trapped at a local minimum point, the rate of reduction should be slow. In this work, the following method to reduce the temperature has been used:

$$T_{i+1} = \gamma T_i \tag{5}$$

where $i = 0, 1, ...$ and $\gamma = 0.99$.

Thus, at the start of MPSOM most worsening moves may be accepted, but at the end only improving ones are likely to be allowed. This can help the procedure jump out of a local minimum.

- **Update Inertial factor**: the inertial factor, used to control relation speed between previous and current speed of each particle, is defined as follows:

$$w_k = w_0 \times \left(1 - \frac{T_0 - T_k}{T_0}\right) \tag{6}$$

where:

  - $w_0 = 0.9$ correspond to the start weight value of the algorithm. Note that with lower value of w the searching region of the algorithm could be around the best solution, and with higher level of w, the algorithm could enhance the exploration searching (global searching).
  - T_0 is an initial temperature and and T_k represents temperature at k^{th} iteration.

- **Mutation operator**: If no improvement of the global best solution does occur during the lasts K iterations, then it means that the algorithm is trapped at a local optimum point. To escape out from local optimum, our MPSOM algorithm uses mutation operator based on the following idea: by given a maximum velocity allowed to each particle, equation 2 ensures that all particles will jumps out from local optimum point and MPSOM algorithm could have a large range of exploration ability.

4. Experiments results on Benchmark functions

To compare the performance of our MPSOM algorithm with those described in [19, 24, 18], we use 7 benchmark functions [7] described in Table 4. These functions possess some properties similar to real world problems and provide a good launch pad for testing the credibility of an optimization algorithm. For these functions, there are many local optima in their solution spaces. The amount of local optima increases with increasing complexity of the functions, i.e. with increasing dimension.

4.1. Comparison with results obtained by [19, 17, 18]

In order to make a fair comparison of classical PSO, Gaussian PSO with jumps algorithm (GPSO-J) [24], Comprehensive Learning PSO algorithm (CLPSO) [18], Particle Swarm Optimizer with Adaptive Tabu and Mutation (ATM-PSO) [19] and our MPSOM approach we fixed, as indicated in [19], the number of particles in the swarm at 20 and the maximum number of function evaluations (FEs) is set to $5000\ D = 150000$. A total of 30 runs for each experimental setting were conducted and the average fitness of the best solutions throughout the run is recorded.

Analysis of numerical results given in Table 2 show that:

- All the algorithms successfully solve Sphere's function which is the simplest unimodal function even if our algorithm obtains best solutions.
- Our MPSOM algorithm outperforms other PSO variants when optimizing hard multimodal problems. In fact, for Rosenbrock's function, which is considered in the optimization literature as difficult problem due to the nonlinear interaction between variables [13], we can see that our algorithm successfully solve this problem while the other algorithms get stuck in local optima.
- ATM-PSO and CLPSO successfully solve Schwefels problem. In contrast, GPSO-J and our MPSOM algorithm fail to optimize this problem.
- For highly multimodal Rastrigins function, ATM-PSO and our MPSOM approach hit the global optimal solution reliably within 150000 FEs.
- The analysis of the results obtained for Ackley's function shows that ATM-PSO and MPSOM obtain better mean results than GPSO-J and CLPSO.
- The optimization of the Griewanks function is an example of the failure of all algorithms except MPSOM that hit the global optimal solution.
- The optimization of the Quartic noisy function is another example of the success of MPSOM and illustrates that other algorithms are adapted to a dynamic environment.

Table 1. Standard benchmark functions adopted in this work.

Function	Problem	Range	$f(x^*)$	ϵ	Classification
Sphere	$\sum_{i=1}^{n} x_i^2$	[-100;100]	0	0.01	Unimodal
Rastrigin	$\sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12;5.12]	0	10	Multimodal
Griewank	$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600; 600]$	0	0.1	Multimodal
Rosenbrock	$\sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[-2.048;2.048]	0	100	Unimodal
Schwefel	$420.9687\, n - \sum_{i=1}^{n}(x_i \sin(\sqrt{\lvert x_i\rvert}))$	[-500.0;500.0]	0	2000	Multimodal
Ackley	$20 + e - 20\, e^{-0.2\,(\frac{1}{n}\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	$[-30.0; 30.0]$	0	0.1	Multimodal
Quadratic	$(\sum_{i=1}^{n}(i+1)x_i^4) + rand[0, 1]$	$[-1.28; 1.28]$	0	0.1	Noisy

Table 2. Mean results and standard deviation over 30 independent runs for 4 PSO variants algorithms on 7 benchmark functions(Dimension $n = 30$ and swarm size = 20).

Function	$CLPSO$	$GPSO - J$	$ATM - PSO$	**MPSOM**
Sphere	8.99e-14	3.85e-07	8.90e-104	**9.4008e-113**
	± 4.66e-14	± 7.97e-07	± 3.18e-103	**± 9.6549e-113**
Rosenbrock	20.88	27.11	15.13	**0.014441893**
	± 2.58	± 16.91	± 8.71e-01	**± 0.016312307**
Schwefel	1.76e-12	1336.29	2.36e-12	**4501.7**
	± 3.27e-13	± 290.81	± 9.57e-13	**± 1520.45862**
Rastrigin	1.34e-06	15.86	0	**0**
	± 1.66e-06	± 4.52	± 0	**± 0**
Ackley	8.45e-08	1.43e-03	2.59e-14	**2.60232e-10**
	± 1.96e-08	± 1.36e-03	± 6.12e-15	**± 5.27812e-11**
Griewank	1.95e-09	4.02e-02	2.22e-02	**0**
	± 4.35e-09	± 4.02e-02	± 2.03e-02	**± 0**
Quadratic	8.18e-03	4.45e-03	9.77e-03	**4.0721e-06**
	± 2.39e-03	±9.95e-04	±3.16e-03	**± 1.055e-05**

Convergence Rate: $Q_{measure}$

In order to evaluate the convergence rate of all algorithms compared in this section, a threshold ϵ is set for each benchmark function as described in [19]. When each algorithm reaches the specified threshold ϵ (see Table 2) of a certain benchmark function in the k^{th} trial, the number of function evaluations FE_k needed is recorded and the current trial k is denoted as a successful trial.

[16] proposes a $Q_{measure}$ criterion which incorporates the measure of both convergence and robustness. The $Q_{measure}$ used, to evaluate performance of all algorithms, is defined as follows:

$$Q_{measure} = \frac{n_t \sum_{i=1}^{n_s} FE_i}{n_s^2} \tag{7}$$

where:

- n_t denote the total number of trials,
- n_s denote the number of successful trials,
- the success ratio SR is defined as $SR = \frac{n_s}{n_t}$.

$Q_{measure}$ values of four algorithms on 7 benchmark functions studied in this work are summarizes in Table3. Analysis of this table show that MPSOM converges slightly faster than all other algorithms, except for the optimization of Schwefel's function where CLPSO and ATM-PSO achieve very good performance in comparison with our MPSOM.

Table 3. $Q_{measure}$ values of four algorithms on 7 benchmark functions studied in this work. Percentage in parentheses are success ratio (SR).

Function	$CLPSO$	$GPSO-J$	$ATM-PSO$	**MPSOM**
Sphere	67977	19343	8101	**216.66**
	(100%)	(100%)	(100%)	(100%)
Rosenbrock	34654	6537	980	**700.66**
	(100%)	(100%)	(100%)	(100%)
Schwefel	24439	47987	9726	**21271**
	(100%)	(100%)	(100%)	(100%)
Rastrigin	87201	1795500	15187	**3248**
	(100%)	(7%)	(100%)	(100%)
Ackley	62659	42857	9664	**4832**
	(100%)	(100%)	(100%)	(100%)
Griewank	65437	18309	7140	**5085**
	(100%)	(90%)	(100%)	(100%)
Quadratic	38094	11137	5988	**1177.33**
	(100%)	(100%)	(100%)	(100%)

5. Reducing Memory energy consumption in embedded systems

Text venir

6. Conclusion

In this paper, we have designed a new algorithm (MPSOM) that uses the exploration ability of PSO and mutation operator to avoid premature convergence. Compared with ATM-PSO [19], CLPSO [18] and GPSO-J [24] on 7 well-known benchmark functions (unimodal, noisy, multimodal) and on problem of reducing Memory energy consumption in embedded systems, it has been shown that our approach has better performances in terms of accuracy, convergence rate, stability and robustness. In future work, we will compare MPSOM algorithm with other hybrid algorithms (GA-SA, PSO-GA) whose design is in progress by the authors. Comparison will also be done on additional benchmark functions and more complex problems including functions with dimensionality larger than 30.

ACKNOWLEDGMENTS

This work is financed by the french national research agency (ANR) in the Future Architectures program.

References

[1] Pant M., Thangaraj R. and Abraham A., *Particle Swarm Based Meta-Heuristics for Function Optimization and Engineering Applications*. 7^{th} Conf. Computer Information Systems and Industrial Management Applications. Vol. 7, pp. 84-90, Issue , 26-28, 2008.

[2] Locatelli M., *Simulated annealing algorithms for continuous global optimization*. In: P.M. Pardalos, H.E. Romeijn (Eds.) Handbook of Global Optimization, Vol. 2, pp. 179-230, Kluwer Academic Pub. 2002.

[3] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Academic Press, 2001.

[4] Marco A. Montes de Oca and Thomas Sttzle. *Convergence behavior of the fully informed particle swarm optimization algorithm*. Proceedings of the 10th annual conference on Genetic and evolutionary computation. pp. 71-78, 2008.

[5] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, Chichester, West Sussex, England, 2005.

[6] R. Poli, J. Kennedy, and T. Blackwell. *Particle swarm optimization. An overview*. Swarm Intelligence, 1(1):3357, 2007.

[7] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*. Technical Report 2005005, Nanyang Technological University, Singapore and IIT Kanpur, India, May 2005.

[8] Sandgren E, *Nonlinear Integer and Discrete Programming in Mechanical Design*, In Proc. of the ASME De-

sign Technology Conference, Kissimme,Fl, pp. 95-105, 1998.

[9] Price W. L., *A Controlled Random Search Procedure for Global Optimization*, In: L. C. W. Dixon and G. P. Szego, eds., Towards Global Optimization 2, North Holland Pub. Company, pp. 71-84, 1978.

[10] Kennedy, J. and Eberhart, R. C., *Particle swarm optimization*. In Proceedings of the IEEE International Conference on Neural Networks, pp. 1942-1948, 1995.

[11] Shi, Y. and Eberhart, R. C. *A modified particle swarm optimizer*. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway, NJ. pp. 69-73, 1998

[12] S. Kirkpatrick and C. D. Gelatt and Jr. and M. P. Vecchi, *Optimization by Simulated Annealing*. Journal of Science, vol. 220, pp. 671-680, 1983.

[13] D. Ortiz-Boyer, C. Herbaś-Martínez, and N. Garciá-Pedrajas. *CIXL2 - A crossover operator for evolutionary algorithms based on population features*. Journal of Artifical Intelligence Research, vol.24: pp. 1-48, 2005.

[14] T. Bäck. *Evolutionary algorithms in theory and practice*. Oxford University, Press, 1996.

[15] Kannan B. K. and Kramer S. N, *An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design*, Journal of Mechanical Design. Vol. 116. pp. 405-411, 1994.

[16] Feoktistov, *Differential Evolution: In Search of Solutions*, Vol. 5, 2006, Springer.

[17] Brest J., Greiner S., Boskovic B., Mernik M., and Zumer V., *Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems*. IEEE Trans. Evol. Comput. Vol. 10, N. 6, pp. 646657, 2006.

[18] Liang J. J., Qin A. K., Suganthan P. N. and Baskar S, *Comprehensive learning particle swarm optimizer for global optimization of multimodal functions*, IEEE Trans. Evol. Comput. Vol. 10, N. 3, pp. 281295, 2006.

[19] Yu-Xuan W., Qiao-Liang X., Zhen-Dong Z., *Particle swarm optimizer with adaptive tabu and mutation: A unified framework for efficient mutation operators*, Journal ACM Transactions on Autonomous and Adaptive Systems (TAAS), Vol. 5 , Issue 1, February 2010.

[20] Geetha T and Michael A, *Effective Hybrid PSO and K-Means Clustering Algorithm for Gene Expression Data*, Journal of Rapid Manufacturing Vol. 1, No.2, pp. 173 188, 2009.

[21] Idoumghar L. and Schott R., *Two Distributed Algorithms for the Frequency Assignment Problem in the Field of RadioBroadcasting*, Journal of IEEE Transactions on Broadcasting, Vol. 55, Issue 2, Part 1, pp. 223-229, June 2009.

[22] Premalatha K. and Natarajan A. M., *Combined Heuristic Optimization Techniques for Global Minimization* Journal of Advances in Soft Computing and Its Applications, Vol. 2, No. 1, pp. 85-99, March 2010.

[23] Ratnaweera A., Halgamuge S., Andwatson H., *Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients*, IEEE Trans. Evol. Comput. Vol. 8, Issue 3, pp. 240255. 2004.

[24] Krohling R. A., *Gaussian particle swarm with jumps*. Proceedings of the IEEE Congress on Evolutionary Computat. pp. 12261231, 2005.