MHPSO: A New Method to Enhance the Particle Swarm Optimizer

BafrinZarei[1] , Reza Ghanbarzadeh[2] , Kiarash Shakibaei[2]
[1]Young Researchers Club
Tabriz Branch, Islamic Azad
University, Tabriz, Iran
[2,3]Department of Computer Engineering
Heris Branch, Islamic Azad University
Heris, Iran
Bafrin.zarei@gmail.com, {ghanbarzadeh, shakibaei}@herisiau.ac.ir

ABSTRACT: *The widespread and increasing application of Particle Swarm Optimizer (PSO) algorithms in both theoretical and practical fields leads to further considerations and new developments for improving its efficiency. To achieve this purpose in this paper a new method is introduced to enhance the convergence rate and reduce the computational time of PSO by combining the PSO including mutation concept (MPSO) and the Hierarchical Particle Swarm Optimizer (HPSO). Therefore the new approach is called MHPSO: a composition of MPSO and HPSO which act simultaneously in the optimization process. In addition some benchmark examples are analyzed using the presented method; consequently, the results are compared to other procedures which illustrate better outcomes and high performance of MHPSO.*

Keywords: Particle Swarm Optimizer, Mutation PSO, Hierarchical PSO, Acceleration Coefficients

Received: 13 May 2011, Revised 20 July 2011, Accepted 27 July 2011

1. Introduction

Nowadays many problems in various scientific fields are getting more complicated and large scale which boils down to extra consideration on their solutions. Although introducing artificial intelligence algorithms leads to more precise results in less computational time, yet further verifications are necessary to make them more efficient. Particle Swarm Optimizer (PSO) is one of the most powerful and simplest tools with a wide range of application in this point. In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster and cheaper way compared by other methods. In computer science, PSOis a computational method optimizing a problem using iterative method to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. PSO is not required for the optimization problem to be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. It optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to the simple mathematical formulae. According to the widespread applications of this algorithm in most mathematical engineering problems, extra studies are required to enhance its performance.

In this paper, first, the basic PSO is reviewed. Then, the mutation concept in MPSO and Hierarchical Particle Swarm Optimizer (HPSO) which were introduced by Ratnaweera, Halgamuge and Watson are described in advanced [1]. Finally, the new

approach that utilizes both MPSO and HPSO methods simultaneously is introduced to develop a high efficient PSO algorithm for optimization problems.

Benchmark examples are studied to show the proficiency of presented method and the results of previous studies are compared by new method.

2. Particle Swarm Optimizer

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [2]. PSO algorithms are especially useful for parameter optimization in continuous, multi-dimensional search spaces. It is mainly inspired by social behavior patterns of organisms that living and interacting within large groups. In particular, PSO incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees.

Although PSO shares many similarities with Evolutionary Computation techniques, basically with Genetic Algorithms and Evolutionary Strategies, there are significant differences with those techniques. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Another attractive feature of PSO is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

In this algorithm each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. How much influence a particular point has on other points is determined by its fitness that is a measure assigned to a potential solution, which captures how good it is compared to all other solution points. Hence, an evolutionary idea of "survival of the fittest" (in the sense of Darwinian evolution) comes into play, as well as a social behavior component through a "follow the local leader" effect and emergent pattern formation. This value is called "*pbest*". Another "*best*" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called "*lbest*". When a particle takes all the population as its topological neighbors, the best value is a global best and is called "*gbest*".

The connection to search and optimization problems is made by assigning direction vectors and velocities to each point in a multi-dimensional search space. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its *pbest* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

PSO algorithm can be explained simply as follows:

1. Initialize the population - locations and velocities
2. Evaluate the fitness of the individual particle (*pBest*)
3. Keep track of the individuals highest fitness (*gBest*)
4. Modify velocities based on *pBest* and *gBest* position
5. Update the particles position
6. Terminate if the condition is met
7. Go to Step 2

Although, PSO is an operational optimization procedure, yet some drawbacks encourage scientists to enhance its performance using several mathematical and technical methods. In this view of point three different developments introduced by Ratnaweera, Halgamuge and Watson in 2004are explained in the next section which are used to present the new proposed method [1].

3. Some previous works

Much as the basic PSO methods are capable of locating a good solution at a significantly faster rate, when compared to other

evolutionary optimization methods, their ability to fine tune the optimum solution is comparatively weak, mainly due to the lack of diversity at the end of the search [3]. Also, in PSO, problem-based tuning of parameters is also a key factor to find the optimum solution accurately and efficiently [4]. Considering these facts, Ratnaweera, Halgamuge and Watson proposed three strategic developments to improve the performance of PSO which can be expressed as complete descriptions, formulations and pseudocodes can be found in the reference [1]. These strategies are explained briefly as following.

3.1 Time-Varying Acceleration Coefficients (TVAC)

In PSO, the search toward the optimum solution is guided by the two stochastic acceleration components, the cognitive component and the social component. Therefore, proper control of these two components is very important to find the optimum solution accurately and efficiently. Considering those concerns, they proposed time-varying acceleration coefficients which have proposed as a new parameter automation strategy for the PSO concept. The objective of this development is to enhance the global search in the early part of the optimization and to encourage the particles to converge toward the global optima at the end of the search. By changing the acceleration coefficients and time, the cognitive component is reduced whereas the social component is increased. With large cognitive and small social components at the beginning, particles are allowed to move around the search space, instead of moving toward the population best. On the other hand, a small cognitive and a large social components allow the particles to converge to the global optima in the latter part of the optimization. This is referred to as PSO-TVAC method.

3.2 Particle Swarm Optimizer With "Mutation" and Time-Varying Acceleration Coefficients (MPSO-TVAC)

In PSO, lack of population diversity, particularly during the latter stages of the optimization, can be understood as the dominant factor for the convergence of particles to local optimum solutions prematurely. Recently, several attempts on improving diversity of the population have been reported in the literature, considering the behavior of the particles in the swarm during the search [5]-[8]. In addition, possible use of the concept of "mutation" in PSO as a performance enhancing strategy has also been investigated [9].Ratnaweera, Halgamuge and Watson combined "mutation" with particle swarm strategy (MPSO), to enhance the global search capability of the particles by providing additional diversity [1]. Mutation is widely used in most evolutionary optimization methods, such as evolutionary programming and genetic algorithms, to guide and enhance of search toward global best solution [3]-[10]-[12].Moreover, global search is enhanced via the introduction of a mutation operator, which is conceptually equivalent to the mutation in genetic algorithms. By using this strategy, when the global optimum solution is not improving with the increasing number of generations, a particle is selected randomly and then a random perturbation (mutation step size), is added to a randomly selected modulus of the velocity vector of that particle by a predefined probability (mutation probability).

3.3 Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients (HPSO-TVAC)

It has been understood that most of the previous empirical developments of PSO are based on either the inertia weight factor method, with a linear varying inertial weight factor, or the constriction factor method [5]-[8]-[13]-[14].

In [1] the novel concept "self-organizing hierarchical particle swarm optimizer(HPSO)" is introduced to provide the required momentum for particles to find global optimum solution, in the absence of the previous velocity term. In this method, previous velocity term is kept at zero, and the modulus of velocity vector of a particle reinitialized with a random velocity (re-initialization velocity). Therefore, in mentioned method,according to the behavior of the particles in the search space, a series of particle swarm optimizers are automatically generatedinside the main particle swarm optimizer, until the convergence criteria is met.

4. New developed method

By considering the efficient performances of MPSO and HPSO reported in [1], a new method is introduced here which is a combination of using mutation concept in PSO and self-organizing hierarchical particle swarm optimizer which will be referred as MHPSO. This is for further enhancements of PSO in various scientific fields which may result in solving more complicated problems in less time comparing MPSO and HPSO.

In the particle swarm algorithm, the trajectory of each individual in the search space is adjusted by dynamically altering of velocity related to each particle, according to its own flying experience and the flying experience of the other particles in the search space. The position vector and the velocity vector of the i^{th} particle in the *d*-dimensional search space can be represented as (1) and (2) respectively. According to a user defined fitness function, let us say the best position of each particle which corresponds to the best fitness value obtained by that particle at time *t*, is as (3), and the fittest particle found so far at time *t* is as (4).

$$\boldsymbol{X}_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{id}) \quad (1)$$

$$\boldsymbol{V}_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{id}) \quad (2)$$

$$\boldsymbol{P}_i = (p_{i1}, p_{i2}, p_{i3}, \ldots, p_{id}) \quad (3)$$

$$\boldsymbol{P}_g = (p_{g1}, p_{g2}, p_{g3}, \ldots, p_{gd}) \quad (4)$$

The proposed pseudocode of MHPSO is as follows:

Begin
Initialize the population
While (termination condition= false)
Do
For (i=1 to number of particles)
Evaluate the fitness:=f(x)
Update P_{id} and P_{gd}
For)d= 1 to number of dimentions(
Calculate the new velocity
$v_{id}= c_1 * rand1(.)*(p_{id}-x_{id})+c_2 * rand2(.)*(p_{gd}-x_{id})$
If(v_{id}=0)
If(Rand3(.)<0.5)
v_{id}=rand4(.)*v
else
v_{id}=-rand5(.)*v
End if
v_{id}=sign(v_{id})*min(abs(v_{id}, v_{max}))
Update the position
Increase d
Increase i
For (i=1 to number of particles)
Evaluate the fitness:=f(x)
Update P_{id} and P_{gd}
For(d= 1 to number of dimentions)
Calculate new velocity := v_{id}
Update the position
Increase d
Increase i
Select a random particle := k
Select a random dimension :=1
If(Δglobal <= 0)
If(rand1(.)<p_m)
If(rand2(.)<0.5)
$V_{kl}= V_{kl} + rand3(.)*v_{max}/m;$
Else
$V_{kl}= V_{kl} – rand4(.)*v_{max}/m;$
End if
End if
End if
Enddo
End
End

Where c_1 and c_2 are constant values named as acceleration coefficients, and rand(.) and Rand(.) are two separately generated uniformly distributed random numbers in the range [0,1]. rand i(.), i = 1,2,3,...,4 are separately generated, uniformly distributed random numbers in the range [0,1], p_m is the mutation probability, Δ global is the rate of improvement of the global solution over the generations and *m*, *k* and *I* are constants. In addition *v* is the re-initialization velocity.

Similar to PSO, MPSO and HPSO, MHPSO can be exploited in different optimization strategies and applications such as Robot path panning, Routing, Image and pattern recognition, topology selection, cloud computing environments, system stability enhancement, identification system, power distribution and so on.

5. Simulation set up

Simulations have been carried out with Matlab® [9] using visual Basic Application (VBA) macro language, which permits an easy implementation of any optimization method. For indicating the efficiency of proposed method, new developed algorithm is examined using a well-known benchmark function called "Griewank Function" which is demonstrated as (5).

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (4)$$

This functionhas the global minimum at the origin or very close to the origin [15]. Simulations were carried out to find the global minimum of the function using PSO, MPSO, HPSO and the new presented method, MHPSO. By selecting the population size equal to 40 in all examples, the simulation results are given in the tables I to IV. Every value in all tables has obtained from average of 20 different runs with the same input parameters in simulations.

During the early stages of the PSO method development, symmetric initialization was widely used, where the initial population is uniformly distributed in the entire search space. Later, Angeline [3] introduced the asymmetric initialization method, in which the population is initialized only in a portion of the search space. In this study the second method is applied where the search space and the initial population are selected in the range of (-600,600) and (300,600), respectively.

It is quite common in PSO methods to limit the maximum velocity of each modulus of the particle velocity vector to a maximum allowable velocity, in order to limit excessive searching outside the predefined search space. Through empirical studies on numerical benchmarks, Eberhart and Shi[16] suggested that it is better to limit the maximum velocity to the upper value of the dynamic range of search. Therefore, this limitation was used for simulation of the new method considering $\mathbf{V}_{\max} = \mathbf{X}_{mx} = 600$.

Table I shows the results obtained from various algorithms with swarm size equal to 30 in 50 iterations whereas number of generations equals to 2000, employing variable c_1 and c_2. Obviously, optimum solution is found by MHPSO when c_1=2-0.75 and c_2=0.75-2. In this simulation mutation probability is calculated from relation (6).

$$P_m = 1 - \left(\frac{t}{T}\right)^{0.95} \qquad (5)$$

In addition, four discussed algorithms are compared in three different conditions with variable swarm size and number of generations. The results are shown in Table II employing c_1=2-0.5 and c_2=0.5-2 in 50 iterations. The mutation probability function is selected as same as the previous one. Table III indicates the obvious impact of re-initialization velocity on simulation employing MHPSO. To show the effect of mutation probability, MPSO and MHPSO methods are examined resulting in Table IV with variable swarm size and number of generation which proves the efficiency of the new presented method.

Finally, the convergence rates are illustrated in Figure 1 and Figure 2 indicating the high performance of MHPSO algorithm in comparison by MPSO. As it can be seen from the diagrams, the proposed method is comparatively fast and efficient three times more than previous methods. It is forecasted that with real world data, further efficient results could be achieved.

6. conclusion

According to some deficiencies in Particle Swarm Optimizer algorithm and its wide range of applications, new developments are

Algorithm		PSO	HPSO	MPSO	MHPSO
c1=1	c2=2	93.2023	0.0605	3.21E-07	1.18E-07
c1=2	c2=1	62.7593	0.0105	2.78E-06	8.92E-07
c1=2	c2=2	19.9797	0.0051	1.68E-05	2.42E-06
c1=2-0	c2=0-2		1.0212	1.17E-07	7.81E-05
c1=2-0.25	c2=0.25-2		1.013	1.70E-07	0.22E-07
c1=2-0.5	c2=0.5-2		0.0868	5.61E-07	6.31E-07
c1=2-0.75	c2=0.75-2		0.0481	1.54E-06	1.13E-07
c1=2-1	c2=1-2		0.0309	9.49E-07	1.64E-07
c1=2.25-0.5	c2=0.5-2.25		0.0145	3.80E-07	3.89E-07
c1=2.5-0.5	c2=0.5-2.5		0.0057	1.19E-06	0.05E-07
c1=2.75-0.5	c2=0.5-2.75		0.0266	1.69E-06	1.18E-07
c1=2.5-0.75	c2=0.75-2.5		0.0115	2.38E-06	8.57E-07

Table 1. Effect of c1 and c2 in various PSO algorithms

Swarm Size	Number of generations	PSO	MPSO	HPSO	MHPSO
10	3000	17.8	3.86E-13	2.04E-14	1.23E-14
20	4000	49.7	3.22E-10	2.13E-08	2.76E-11
30	5000	81.4	2.32E-10	2.09E-04	2.84E-09

Table 2. Comparison of different PSO algorithms considering various swarm sized and nomers of generations

Swarm Size	Numberof generations	50% Vmax	20% Vmax	10% Vmax	5% Vmax	50% Vmax
10	3000	7.23E-13	7.12E-13	7.13E-13	7.43E-13	7.13E-13
20	4000	2.76E-10	2.56E-10	2.41E-10	2.36E-10	2.51E-10
30	5000	2.84E-09	2.51E-09	2.57E-09	2.52E-09	2.59E-09

Table 3. The effect of re-initialization velocity

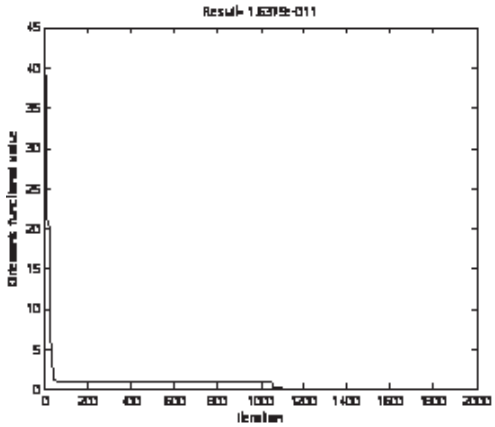

Figure 1. Convergence time-history of MHPSO

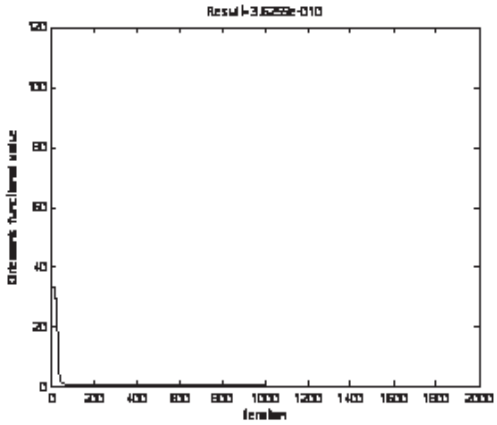

Figure 2. Convergence time-history of MPSO

Probability of Mutation	Swarm Size	Number of generations	….. Result	
			MPSO	MHPSO
1	10	3000	0.06	9.83E-01
	20	4000	0.036	5.25E-02
	30	5000	0.025	0.0424
0.8	10	3000	0.0238	0.0483
	20	4000	0.0166	0.0231
	30	5000	0.0105	0.0137
0.6	10	3000	0.0099	0.0073
	20	4000	0.0044	0.0063
	30	5000	0.002	0.0055
0.4	10	3000	5.94E-04	0.0011
	20	4000	2.53E-04	6.96E-04
	30	5000	3.20E-04	6.50E-04
0.2	10	3000	5.84E-06	1.03E-05
	20	4000	3.30E-06	1.90E-06
	30	5000	2.54E-06	7.02E-06

Table 4. The effect of mutation probability

inevitable in this field. Therefore, a new method is presented which is a combination of mutation based PSO algorithm (MPSO) and the hierarchical PSO (HPSO). Considering these two algorithms which were developed previously, the new one is referred as MHPSO. Simulation results concerned with proposed method comparing the previous solutions, illustrates the efficiency of MHPSO.

References

[1] Ratnaweera, A., Halgamuge, S. K., Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *In:* Proc. of the IEEE Trans. Evol- Comput, 8 (3) 240–255.

[2] Kennedy, J., Eberhart,R. (1995). Particle swarm optimization. *In:* Proc. of the IEEE International Conference on Neural Networks, IEEE Press, Piscataway, NJ, p. 1942-1948.

[3] Angeline, P. J. (1998). Evolutionary optimization verses particle swarm optimization: Philosophy and the performance difference. *In:* Lecture Notes in Computer Science, V. 1447, Proc. 7th Int. Conf. Evolutionary rogramming Evolutionary Programming VII, p. 600–610.

[4] Shi, Y., Eberhart,R. C. (1998). Parameter selection in particle swarm optimization. *In:* Lecture Notes in Computer Science—Evolutionary Programming VII, V. 1447, Proc. 7th Int. Conf. Evolutionary Programming, p. 591–600.

[5] Lovbjerg, M., Krink,T. (2002). Extending particle swarm optimizer with self-organized critically. *In*: Proc. of the IEEE Int. Congr. Evolutionary Computation, V. 2, Honolulu, HI, p. 1588–1593.

[6] Vesterstrom, J. S., Riget, J., Krink, T. (2002). Division of labor in particle swarm optimization. *In:* Proc. of the IEEE Int. Congr. Evolutionary Computation 2002, V. 2, Honolulu, HI, p. 1570–1575.

[7] Krink,T., Vesterstrom, J. S., Riget,J. (2002). Particle swarm optimization with spatial particle extension. *In:* Proc. of theIEEE Congr. Evolutionary Computation, V. 2, Honolulu, HI, p. 1474–1479.

[8] Xie, X. F., Zhang, W. J., Z.-L.Zhi-Lian Yang, (2002). A dissipative particle swarm optimization. *In:* Proc. of theIEEE Congr. Evolutionary Computation, V. 2, Honolulu, HI, p. 1456–1461.

[9] Higashi, N., Iba, H. (2003). Particle swarm optimization with Gaussian mutation. *In:* Proc. of theIEEE Swarm Intelligence Symposium (SIS 2003), Indianapolis, IN, p. 72–79.

[10] Shi,Y., Eberhart,R. C(1998). Comparison between genetic algorithms and particle swarm optimization. *In:* Lecture Notes in Computer Science Evolutionary Programming VII, V. 1447, Proc. 7th Int. Conf. Evolutionary Programming, p. 611–616.

[11] Eberhart, R. C., Shi,Y. (2001). Tracking and optimizing dynamic systems with particle swarms. *In:* Proc. of the IEEE Congr. Evolutionary Computation 2001, Seoul, Korea, p. 94–97.

[12] Yoshida, H., Kawata, K., Fukuyama, Y., Nakanishi,Y. (1999). A particle swarm optimization for reactive power and voltage control considering voltage stability. *In:* Proc. of the Int. Conf. Intelligent System Application to Power System, Rio de Janeiro, Brazil, p. 117–121.

[13] Lovbjerg, M., Rasmussen, T. K., Krink, T. (2001). Hybrid particle swarm optimizer with breeding and subpopulation. *I n:* Proc. of the 3rd Genetic volutionary Computation Conf. (GECCO-2001), San Francisco, CA, p. 469–476.

[14] van den Bergh, F., Engelbrecht, A. P. (2001). Effect of swarm size on cooperative particle swarm optimizers. *In:* Proc. of the Genetic Evolutionary Computation Conf. (GECCO-2001), San Francisco, CA, p. 892–899.

[15] Shi, Y., Eberhart, R. C. (1999). Empirical study of particle swarm optimization. *In:* Proc. of the IEEE Int. Congr. Evolutionary Computation, V. 3, p. 101–106.

[16] Eberhart, R. C., Shi,Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. *In:* Proc. of the IEEE Int. Congr. Evolutionary Computation, V. 1, p. 84–88.