PARTICLE SWARM OPTIMIZATION (PSO) VARIANTS WITH TRIANGULAR MUTATION

R. Hashim[1], M. Imran[1], N. E. A. Khalid[2]

[1]Universiti Tun Hussein Onn Malaysia

[2]Universiti Teknologi MARA Malaysia

ABSTRACT

Particle swarm optimization (PSO) is a stochastic algorithm used for the optimization problems proposed by Kenned in 1995. It is a very good technique for optimization problems. However, there is a drawback - it stuck in the local minima. To improve the performance of PSO, many researchers proposed different variants of PSO. Some of the efforts include improving the initialization of the swarm, introduce new parameters - constriction coefficient and inertia weight, define the different method of inertia weight to improve the performance of PSO, and modifying the global and local best particles by introducing the mutation operators in the PSO. In this paper, we will see the different variants of PSO with respect to initialization, inertia weight and mutation operators. We also proposed a new PSO technique using triangular mutation. The new technique is tested on the benchmarked functions. The results show better performance when compared to four previous PSO variants.

KEYWORDS*: PSO, Overview of PSO, PSO Variants,PSO and mutation Operators, PSO and Inertia Weight*

1.0 INTRODUCTION

PSO is a Mehta heuristic algorithm originally proposed by Kennedy and Eberhart (Kennedy, J., & Eberhart, R., 1995). The algorithm simulates the behavior of bird flock flying together in multi-dimensional space in search of some optimum place, adjusting their movements and distances for better search (Kennedy, J., & Eberhart, R., 1995). PSO is an evolutionary computation method similar to the Genetic Algorithm (GA). Swarms called particles are initialized randomly and then search for optimal by updating generations. PSO has two approaches: one is called cognitive and another is called social.

Corresponding author email: radhiah@uthm.edu.my

The algorithm mimics a particle flying in the search space and moving towards the global optimum. A particle in PSO can be defined as $P_i \in [a, b]$ where i=1, 2, 3…. D and a, b$\in R$, D is for dimensions and R is for real numbers [30]. Each particle has its own velocity and position which are randomly initialized in the start. Each particle has to maintain its positions pbest known as local best position and the G_{best} known as global best position among all the particles. Following equations are used to update the position and velocity of the particle.

$$V_i(t+1) = V_i(t) + C_1 * r_1\left(P_{best} - n_i(b)\right) + C_2 * r_2(g_{best} - x_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

Where V_i is the velocity, X_i is the position P_{best} is the personal best position of the particle and g_{best} is the global best position for the PSO. r_1 , r_2 are two random numbers ranges [0, 1] and C_1 & C_2 are the leaning factors.

2.0 ORIGINAL PSO PSEUDO CODE

According to the PSO algorithm, the objective function that is being optimized will evaluate each candidate solution at each iteration and works on the resulted fitness value. It will maintain several candidate solutions in the search space. To improve the performance of PSO, researchers modified the PSO in different ways. The original PSO pseudo code is shown below:

```
Initialize the population randomly
While (Population Size)
{
Loop
Calculate fitness
If fitness value is better from the best fitness value
(pbest) in history then
Update pbest with the new pbest
End loop
Select the particle with the best fitness value from all
particles as gbest
While maximum iterations or minimum error criteria is not
attained
{
For each particle
Calculate particle velocity by equation (I)
Update particle position according to equation (II)
Next
}
}
```

3.0 ELEMENTS USED IN PSO

Before working with the PSO, we have to know about the elements used in the PSO. First of all, we shall overview the brief concepts of the PSO elements.

- **Particle**---We can define the particle as $P_i \varepsilon[a, b]$ where i=1, 2, 3…D and a, b ε R. Here D is for dimension and R is for real numbers.
- **Fitness Function**---Fitness Function is the function used to find the optimal solution. Usually it is an objective function.
- **Local Best**---It is the best position of the particle among all positions visited so far.
- **Global Best**---The position where the best fitness is achieved among all particles visited so far.
- **Velocity Updat**e---Velocity is a vector to determine the speed and direction of the particle. Velocity is updated by the equation (1).
- **Position Update**---All particles try to move toward the best position for optimal fitness. Each particle in PSO updates their positions to find the global optima. Position is updated by equation (2). Flow chart of the basic PSO is shown below.

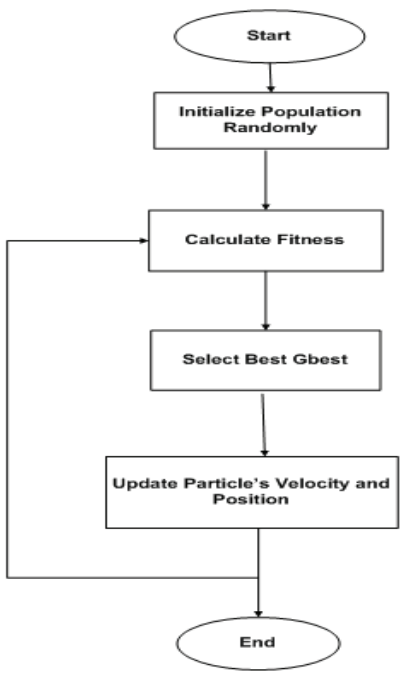

Figure 1. Flow chart of Basic PSO

4.0 PSO VARIANTS

In 1995, Kennedy proposed PSO (Kennedy, J., &Eberhart, R., 1995). As it is mathematically proved so researchers are trying to improve the PSO. Therefore, PSO has different variants with different parameters like, Initialization, Inertia weight and many other techniques. The details of some PSO variants are given below.

4.1 Initialization

Initialization of the particles has an important role in the performance of PSO. If the initialization is not good then algorithm may search in unwanted area and it will be hard to search for the optimal solution. The performance of PSO heavily depends on the initialization of the swarms (Engelbrechr, A. P., 2005). In this section we will see study the different variants of PSO with respect to initialization.

Nguyen et al. (2007) used some low discrepancy sequence to initialize the particles. Researcher used Halton, Sobol, and Faure sequences to initialize the swarm. They test their proposed variants using six standard benchmark functions. They found that the performance of PSO with sobol initialization is best among all the techniques.

Jabeen et al. (2009) presented an opposition based PSO. Author claimed that by the social phenomena if one person is bad then his opponent is good. They generate the population and opposite population then calculate the fitness; the fitter one population is selected to run the PSO. According to author the opposite particle is defined as

$P_{opi} = a+b-p_i$ where i=1, 2, 3…. D is dimension and a, bεR
Where D is the dimension and R is real number.

Pant et al. (2008) used qasi random sequence for initialization of the swarm to improve the PSO performance. The authors used the vndercorput and sobol sequence for the swarm initialization. They used four benchmark functions to perform the experiment. According to the results presented by the author, the performance of VC-PSO used vendor corrupt sequence and SO-PSO used sobol sequence remain dominant on the simple PSO.

Omran (2008) proposed an opposition based learning to improve the performance of PSO. They propped three variant of PSO. In one experiment, they generated the opposite particles and then calculate the fitness of both particles before selecting the fitter particles to run

the PSO. In another variant, the lowest fitter particle was replaced with its opposite particle during each iteration. Eight benchmark functions ware used to test the performance of propped variants with the basic PSO. The results show that the presented variants perform well.

Zhang et al. (2009) proposed an enhance variant of PSO which they called quasi-oppositional comprehensive learning particle swarm optimizers (QCLPSO). They used the qausi opposite numbers for the swarm initialization.

4.2 Constriction Factor

In this section, we will just overview a variant of PSO in which constriction factors has been introduced.

Clerc (2002) proposed an approach to balance the exploration and exploitation by introducing a new parameter 'χ' called constriction factor. By Clerc following equation used to update the velocity

$$v_{ij}(t+1) = \chi\left[v_{ij}(t+1) + \varphi_1\left(v_{ij}(t) - x_{ij}(t)\right) + \varphi_2(\hat{y}_{ij}(t) - x_{ij}(t))\right] \quad (3)$$

where $\chi = 2 \div \left|4 - \varphi - \sqrt{\varphi 2 - 4\varphi}\right|$

$\varphi = \mathrm{C1} + \mathrm{C2}, \varphi_1 = c_1 r_1, \varphi_1 = c_1 r_1$

4.3 Inertia Weight

Inertia weight is an approach like the constriction coefficient to balance the exploration-exploitation trade off. The bigger value of inertia weight encourages the exploration and smaller value of inertia weight encourages for the exploitation. Shi (1998) first time introduce the inertia weight to control the exploration and exploitation. Some researcher used fix inertia weight, some used linearly decreasing and some used non-linearly decreasing inertia weight. This section discuss about the variations of basic PSO by modifying the inertia weight in different ways.

Li (2009) used the exponent decreasing inertia weight. To balance the local and global search abilities the author presented the exponent decreasing inertia weight as

$$w = (w_{ini} - w_{end} - d_1)\exp\left(\frac{1}{1 + d_2 t / t_{max}}\right) \quad (4)$$

where t_{max} shows the max iteration, t shows the t^{th} iteration, w_{ini} shows

the actual inertia weight,w_{end} shows the inertia weight value when the algorithm process run the max iterations, d_1 and d_2 is a factor to control w between w_{ini} andw_{end}. Chongpeng et al. (2007) proposed a variant of PSO with non-linearlydecreasing inertia weight as

$$w_0 = w_{end} + (w_{end} - w_{start}) \times (1 - (t/t_{max})^{k_1})^{k_2} \quad (5)$$

where k_1, k_2 are two natural numbers, w_{start} is the initial inertia weight, w_{end} is the final value of weighting coefficient, t_{max} is the maximum number of iteration and t is current iteration. Value of k_1>1 and K_2=1. Shi (2001) used fuzzy set of rules to adjust inertia weight dynamically.

Zhang et al. (2003) generate a uniformly distributed random number between [0,1]. Author claimed that the performance of proposed inertia weight is much better than the linearly decreasing inertia weight. Moreover they said that through their technique two draw backs of linearly inertia weight can be overcome. One the dependency of inertia weight on the maximum number of iteration the another one is avoiding the lacks of local search ability in the start and global search ability at the end.

Wei et al. (2006) introduce a new version of inertia weight; inertia weight changed dynamically based on speed and accumulation factors.

Pant et al. (2007) used Gaussian distribution to set the inertia weight. Following equation used to set the inertia weight

$$w = abs(rand)/2 \quad (6)$$

where rand is random number generated by Gaussian distribution. Xue dan Liu et al. (2009) used following equation to set the inertia weight

$$w(t) = .9 - \left(\frac{t}{max_t}\right) \times 0.5 \quad (7)$$

4.4 Mutation Operators

To improve the performance of PSO and to escape it from the local minima, the researches proposed different variants of PSO with mutation operators. Some researchers mutate the global best particle and some mutate the local best particle with different techniques to prevent the PSO for stagnation in local minima. Here we will discuss

some of the PSO variants with respect to mutation operators.

Wang et al. (2007) proposed a variant of PSO with Cauchy mutation. Author mutates the global best particle then compares its fitness with the original particle fitness, the fitter one is selected.
Following equation used to mutate the particle.

$$gbest_i(i) = gbest_i(i) + W(i) * N(x_{min}, x_{max}) \quad (8)$$

where N is a Cauchy distributed function with scale parameter t=1, $N(x_{min},x_{max})$ is a random number between (x_{min},x_{max}) of defined domain of test function and

$$W(i) = \left(\sum_{j=1}^{popsize} V[j][i] \right) / popsize$$

where $V[j][i]$ is the i^{th} velocity vector of jth particle in the population, popsize used for population.

Wang et al. (2007) proposed another variant of PSO in which Cauchy mutation is used with opposition based PSO. Pant et al.(2008) presented a new version of PSO with adaptive mutation. They proposed two versions of PSO AMPSO1 and AMPSO2, in one they mutate the global best particle while in another local best particle was mutated. Following equation used to mutate the particle.

$$g_{best} = g_{best} + \sigma' * Betarand() \quad (9)$$

where $\sigma' = \sigma * \exp(\tau N(0,1) + \tau' N_j(0,1))$, $N(0,1)$ is a normally distributed function with mean zero and standard deviation one, $N_j(0,1)$ is another random number generated for each value of j, τ and τ' are set as $\frac{1}{\sqrt{2n}}$ and $\frac{1}{\sqrt{2\sqrt{n}}}$ respectively and value of σ is originally set as 3. Betarand () is a random number generated by beta distribution with parameter less than 1. Pant et al. (2008) presented another extension of PSO using sobol mutation. They proposed two version of PSO: first one is SM-PSO1 and another one is SM-PSO2. In the first one, they mutate the best particle in the swarm. But in the second one, they work on the worst particle in the swarm. The proposed operator was defined as:

$$SM = R_1 + (R_2)/lnR_1 \quad (10)$$

where R_1 and R_2 are random numbers generated by sobol sequence.

Wu et al. (2009) proposed a variant of PSO by using power distribution. Author applies the power mutation on the global best particle and then check the fitness of both particles original and mutated one and select the fitter one.

Imran et al. (2010) proposed another power mutation operator for opposition based PSO. In their proposed technique, they initialize the PSO with opposite swarms, and then apply the power mutation on global best particle. Author claimed that in this way two times performance of PSO improved; at the initialization and also by mutating global best to prevent PSO from stagnation.

Imran et al. (2011) presented another variant of PSO by introducing the student T mutation. Author used the student T distribution to mutate the global best particle. They claimed that their work has performance over Cauchy mutation and adaptive mutation.
Chen (2011) proposed another variant of PSO with mutation operator. In this study, author first generate the mutant particle on the basis of probability, and then check the fitness of both particle and mutant particle, then select fitter one.

5.0 PROPOSED PSO MODIFICATION

From above study, it has been observed that mutating the global best particle using different distribution cause the performance of PSO to improve. However,more investigation to prevent PSO from stagnation in local minima is needed. Therefore author present new versions of PSO, TPSO. In TPSO global best particle is mutated.

The global best particle is mutated as.

$$g_{best} = \begin{cases} g_{best} + t, & g_{best} is + ve \\ g_{best} - t, & g_{best} is - ve \end{cases} \quad (11)$$

where $t = \frac{x^l}{x^u} * STrand()$

where x^l, x^u, are the boundaries of the current search space and $STrand()$ is the random number generated by Triangular distribution.

6.0 BENCHMARK FUNCTIONS

Table 1. Benchmark functions

Function	f_{min}
$f_1(x) = \sum_{i=0}^{n} x_i^2$	0
$f_2(x) = \sum_{i=0}^{n} i * x_i^2$	0
$f_3(x) = \sum_{i=0}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	0
$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0
$f_5(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)^2]$	0
$f_6(x) = \max\|x_i\|, \quad 0 \le i \le n$	0
$f_7(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$	0

7.0 EXPERIMENTAL SETTING

The following experimental setting were used for all techniques:

Table 2. Experimental setting

Parameter	**Value**
Search Space	[100,-100]
C_1=C_2	1.498
Inertia Weight	Linear decrease (0.9-0.4)
Dimensions	10 20 30
Iterations	1000 1500 2000
Population size	30
Number of PSO Runs	30

Keeping above experimental setting we take the average of best 20 runs and the results are displayed in the Table 3.

Table 3. Average of best 20 runs

Function	Dim	Iteration	PSO[1]	OCPSO[17]	AMPSO[19]	STPSO[23]	TPSO
f_1	10	1000	5.03E-53	1.73E-64	4.18E-53	8.98E-98	**6.65E-110**
	20	1500	4.45E-17	3.13E-27	7.37E-17	2.96E-90	**3.55E-101**
	30	2000	1.79E-07	2.39E-14	1.66E-10	7.09E-76	**6.10E-80**
f_2	10	1000	1.36E-65	1.39E-72	1.27E-67	2.33E-80	**2.80E-96**
	20	1500	2.43E-18	1.10E-27	2.19E-18	1.97E-66	**1.83E-73**
	30	2000	1.74E-07	8.75E-14	1.53E-09	1.06E-43	**1.82E-48**
f_3	10	1000	5.37E+00	5.67E+00	5.97E+00	2.14E+00	**1.16E-10**
	20	1500	3.51E+01	3.21E+01	3.64E+01	1.62E+01	**1.94E-15**
	30	2000	9.12E+01	8.60E+01	8.72E+01	3.59E+01	**2.63E-10**
$f4$	10	1000	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	20	1500	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
	30	2000	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**	**1.00E+00**
F_5	10	1000	2.41E+00	1.81E+00	1.10E+00	1.33E+00	**2.65E-8**
	20	1500	6.09E+00	6.09E+00	**4.58E+00**	5.31E+01	3.55E+08
	30	2000	2.95E+01	**2.27E+01**	4.32E+01	8.19E+01	1.15E+05
f_6	10	1000	2.27E-12	**5.12E-17**	4.10E-12	6.19E-16	5.44E-10
	20	1500	2.07E-01	1.25E-02	1.18E-01	3.12E-02	**1.88E-09**
	30	2000	4.72E+00	1.76E+00	4.41E+00	6.75E-01	**2.38E-05**
f_7	10	1000	1.33E-03	1.23E-03	1.08E-03	1.77E-06	**3.67E-09**
	20	1500	8.24E-05	1.59E-04	1.59E-04	1.33E-07	**1.21E-13**
	30	2000	1.50E-05	1.99E-05	1.31E-05	1.56E-09	**4.58E-15**

8.0 CONCLUSION

There are so much work that has been carried out to improve the performance of PSO. In original PSO, there was no inertia weight but to improve the performance, researchers introduced inertia weight. Then, they tried to improve the performance by trying the different initialization methods. Researchers also work on the global best particle to escape from the local minima. For this purpose, they introduce the different mutation operators to improve the performance of PSO.

From above given results, it is observed that the performance of TPSO is significantly better than PSO, OCPSO and AMPSO in function f_1, f_2 and f_3. The performance of all techniques remains the same for function f_4. The fitness of f_5 varies - when less number of dimensions and iterations, OSTPSO performed well.When dimensions are kept 20 with 1500 iterations, AMPSO performed well but when dimensions and iteration are increased to 30 and 2000 respectively, performance of CPO was better. For function f_6, performance of OCPSO was slightly better than STPSO with less number of iterations but when the number of iterations and dimension were increased, TPSO performed well than other techniques. For function f_7, the performance of TPSO is best compared to all other techniques.

Over all we have 21 cases. In 14 cases, TPSO performed better than all other techniques while in 3 cases, performance of all techniques remains the same. OCPSO performed slightly better in 2 cases and AMPSO performed well in just one case.

From above experiments and results, we can say that the presented technique performance better then (Kennedy, J., & E berhart, R., 1995; Wang, H. et al., 2007;Pant, M. et al., 2008; Imran, M. et al., 2011)

9.0 ACKNOWLEDGEMENT

The authors would like to thank University Tun Hussein Onn Malaysia for supporting this research under the Postgraduate Incentive Research Grant Vote No 1031.

10.0 REFERENCES

Chen, L. (2011). Particle swarm optimization with a novel mutation operator. *In International Conference on Mechatronic Science, Electric Engineering and Computer, Jilin*, (970 – 973).

Clerc, M., & Kennedy, J. (2002). The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, *6*, 58–73.

Engelbrechr, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons.

Huang Chongpeng, Zhang Yuling, Jiang Dingguo & XuBaoguo. (2007). On some non-linear decreasing inertia weight strategies in particle swarm optimization. *In Proceedings of the 26th Chinese Control Conference*, Zhangjiajie, Hunan, China, (570-753).

Imran, M., Jabeen, H., Ahmad, M., Ababs, Q., Bangyal, W., &Ababs, Q. (2010). Opposition based PSO and mutation operators (OPSO with Power Mutation). *In 2nd International Conference on Education Technology and Computer*, Shanghai, (V4-506 -508).

Imran, M., Manzoor, Z., Ali, S., & Ababs, Q. (2011). Modified particle swarm optimization with student T mutation. *In International Conference on Computer Networks and Information Technology,* Abbottabad, (283 – 286).

Jabeen, H., Jalil, Z., &Baig, A. R. (2009). Opposition based initialization in particle swarm optimization. I*n Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, New York, USA, (2047- 2052).

Kennedy, J., & Eberhart, R. (1995).Particle swarm optimization. *In Proceedings of IEEE International Conference on Neural Networks* (1942-1948).

Li, H. R., & Gao, Y.L. (2009). Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation. *In Second International Conference on Information and Computing Science*, Manchester, UK, (66-69).

Liu, X. et al. (2009). Particle swarm optimization with dynamic inertia weight and mutation. *In Third International Conference on Genetic and Evolutionary Computing*, Guilin, (620-623).

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

Omran, M. G. H., & al-Sharhan, S. (2008). Using opposition-based learning to improve the performance of particle swarm optimization. *IEEE Swarm Intelligence Symposium*, (1 – 6).

Pant, M., Thangaraj, R., Grosan, C., & Abraham, A. (2008). Improved particle swarm optimization with low-discrepancy sequences. *In IEEE Cong. on Evolutionary,* Hong Kong, (3011-3018).

Pant, M., Thangaraj, T., & Singh, V.P. (2007). Particle swarm optimization using gaussian inertia weight. *In International Conference on Computational Intelligence and Multimedia Applications,* Sivakasi, Tamil Nadu, (97-102).

Pant, M., Thangaraj, R., & Abraham, A. (2008). Particle swarm optimization using adaptive mutation. *In 19th International Conference on Database and Expert Systems*, Washington, DC, USA, (519-523).

Pant, M., Thangaraj, R., Singhand, V.P., & Abraham, A. (2008). Particle swarm optimization using sobol mutation. *In First International Conference on Emerging Trends in Engineering and Technology,* Nagpur, Maharashtra, (367-372).

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *In Evolutionary Computation Proceedings IEEE World Congress on Computational Intelligence,* (69–73).

Shi, Y. & Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. I*n Proceedings of the IEEE Congress on Evolutionary Computation*, Seoul , South Korea, (101-106).

Wang, H. et al. (2007). A hybrid particle swarm algorithm with cauchy mutation. *In Proceeding of IEEE Swarm Intelligence Symposium*, (356 – 360).

Wang, H. et al. (2007). Opposition-based particle swarm algorithm with cauchy mutation. *IEEE Congress on Evolutionary Computation*, (4750 – 4756).

Wei, J. & Wang, Y. (2006). A dynamical particle swarm algorithm with dimension mutation. *International Journal of Computer Science and Network Security,* 6, 221-224.

Xiaoling Wu, Xiaojuan Zhao (2009). Particle swarm optimization based on power mutation. *In International Colloquium on Computing, Communication, Control, and Management, Sanya*, (464 – 467).

Zhang, C., et al. (2009).A novel swarm model with quasi-oppositional particle. *International Forum on Information Technology and Applications*, (325 – 330).

Zhang, L., Yu, H., & Hu, S. (2003). A new approach to improve particle swarm optimization. *In Proceedings of the International Conference on Genetic and Evolutionary Computation*, (134-139).