International Journal on Advanced Science Engineering Information Technology

Vol.7 (2017) No. 1
ISSN: 2088-5334

A New Initialization Technique in Polar Coordinates for Particle Swarm Optimization and Polar PSO

Moaath Shatnawi[#], Mohammad Faidzul Nasrudin[#], Shahnorbanun Sahran[#]

[#]*Center for Artificial Intelligence Technology (CAIT), Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor, Malaysia.*
E-mail: moaathshatnawi@gmail.com, mfn@ukm.edu.my, shahnorbanun@ukm.edu.my

***Abstract*— Particle Swarm Optimization (PSO) is one of the famous algorithms inspired by the natural behavior of a swarm (particles). However, it is used to solve n-dimensional problems in search space. One of its modified versions a Polar Particle Swarm Optimizer was operated in polar coordinates by using an appropriate mapping function introduced based on polar coordinates. The modified algorithm faced some problems, such as generating a distorted search space, which may have been caused by the method of randomization. This paper introduces an initialization technique that operates entirely in polar coordinates. Moreover, an investigation based on standard PSO was done to test the proposed technique. The second part was to use the new initialization technique to enhance the polar PSO performance. In addition, the proposed techniques show evenly distributed points in the polar search space. Furthermore, experimental results were obtained by using various benchmark test functions on different settings of dimensions. While its shows a little enhancement in some benchmark test functions in both PSO and polar PSO, statistically there are no significant differences by using the analysis of variance (ANOVA).**

***Keywords*— polar coordinates; particle swarm optimization; random initialization**

I. INTRODUCTION

PSO and Evolution Strategy (EA) are examples of nature-inspired optimization algorithms [1]. Moreover, PSO was inspired by the flocking and shoaling behavior of birds and fish and was introduced by Kennedy and Eberhart to present the simulation of this behaviour as a new model of computational intelligence field [2]. Furthermore, PSO algorithm was designed as a simple, easy and efficient algorithm to solve several n-dimensional continuous optimization problems present in Cartesian space. The polar evolution strategy (PES) is one of the modified algorithms that operate in polar coordinates to solve the combinatorial problem. This algorithm categorized as one of the differential evolution algorithms [3]. PES is done by following its phenomena, which implies that the region of search will be further away from the origin point with increasing in the number of generations. However, it is crucial while searching in projections or unit vectors to keep the norm fixed. In addition, two applications were applied by PES, which is a spam filtering problem and a credit approval problem [1].

Nevertheless, PSO algorithm modified by changing the fundamental logic such as Binary PSO (*BinPSO*) [4], also another modify is done based on the choosing of the appropriate mapping function. As an example of modified algorithms in polar coordinates is the Polar Particle Swarm Optimizer. The original particle swarm optimization was redefined by using an appropriate mapping function to search in polar coordinates, whereas the fundamental approach of PSO is totally different compare with Evolution Strategy. Thus, growing evidence has also revealed that the experimental results of the modified version of PSO worse than those of its counterpart in Cartesian space [5]. Moreover, the reason of these unsuccessful results happened because of the distortion of search space, which occurred due to the difficulty with floating points when using the transformation equation from polar coordinate to Cartesian coordinates [5]. Therefore, the initialization of particle positions in polar coordinates has recently been utilized, where the diversity of polar initialization differ comparing with Cartesian initialization way. The initialization method in polar coordinates that is used during the modification process of PSO is not diverse enough to cover the polar search space which investigated in [6]. This paper introduces a distribution initialization technique in polar coordinates, which will be used to investigate PSO algorithm behaviour and compare its results with the Cartesian initialization version with PSO. Furthermore, the same technique will be applied to enhance the performance of polar PSO, which will be compared with the previous initialization of PSO in polar

coordinates. However, different scenarios will apply to cover all possible cases of initialization in polar coordinates. In addition, in this research, we intend to use a random distribution and normally truncated distribution to investigate our assumptions. Section II shows the methods, while the initialization in polar coordinates will explain in section III, section IV presents the results, and section V illustrates the conclusion and discussion.

II. Material and Method

This section provides a comprehensive overview of PSO and the polar particle swarm optimizer. Section II-A reviews the original PSO, Section II-B shows the Polar PSO, while section II-C shows the used initialization technique.

A. Particle Swarm Optimization

The PSO algorithm is categorized as a population based algorithm that contains a swarm of particles that are distributed in n-dimensional space [7]. Each particle position represents the possibility of the solution for the optimization problem by finding the fitness value of each particle, where it is used to identify the personal best position [8]. Through the algorithm, two kinds of variant topologies have been defined, which are *lbest* and *gbest*. The global best and local best positions were used to help the particles to search more effectively in the search space by discovering promising regions that will show the way to further exploration where all of this information is shared with the rest of particles. Let n be the size of population (swarm). Each particle i represented with various characteristics were shown as follows:

xi: the current position of the particle;

vi: the current velocity of the particle;

yi: the personal best position of the particle;

Based on the above characteristics of particle i, the velocity of each particle will calculate as follows by use this rule:

$$v_{i,j}(t) = wv_{i,j}(t-1) + \alpha_1(t)(y_{i,j}(t) - x_{i,j}(t)) + \alpha_2(t)(\hat{y}_{i,j}(t) - x_{i,j}(t)) \quad (1)$$

For all dimensions, where w is the inertia weight that defines the effect of its previous velocity on the current velocity; $\alpha_1(t)$ and $\alpha_2(t)$ are defined as $\alpha_1(t)=c_1.r_1(t)$ and $\alpha_2(t)=c_2.r_2(t)$, where $r_1(t)$ and $r_2(t)$ are random values (0,1) and c_1 c_2 is the acceleration contestant used to determine the effect of personal best position $y_{i,j}(t)$ and global best position on the new velocity value. The new position of the particle will update by using the previous equation and the current position of the particle, with the update equation presented as follows:

$$x_{i,j}(t) = x_{i,j}(t-1) + v_{i,j}(t) \quad (2)$$

B. Polar PSO

Polar PSO is one of the latest modified versions of PSO, which was proposed by Matthysen and Engelbrecht [5]. Particle swarm optimization originally operated by producing solution vectors in Cartesian space. The new modified version is operated in polar search space by using an appropriate mapping function. However, the authors addressed some complications of using the polar coordinates, such as that the new search space became a distorted version of the original search space and local optima became enlarged and lost in the global optimum point, which can be seen near the origin. Additionally, The enlargement of the local optimum region which is close to the Cartesian origin is the reason for that distortion, while the global optimum regions reduced in size and got further away.

C. Polar Conversion Function

Polar coordinates offer a good method to represent the points that are presented in Cartesian space as a conversion method. In addition, the polar coordinates system is a good mapping function to present in mathematics, engineering, physics, and robotics, as well as various scientific fields. To allow the PSO algorithm to operate in polar search space, it was required to define a mapping conversion function in the n-dimensional polar vector to be used to convert back to Cartesian space. This was formulated by Kendall [9] as follows:

$$\begin{aligned}
&\vec{x} = \mu(\vec{\theta}) \\
&x_1 = r.\sin(\theta_1).\sin(\theta_2)\dots\sin(\theta_{n-2}).\cos(\phi) \\
&x_2 = r.\sin(\theta_1).\sin(\theta_2)\dots\sin(\theta_{n-2}).\sin(\phi) \\
&x_3 = r.\sin(\theta_1).\sin(\theta_2)\dots\cos(\theta_{n-2}) \\
&\vdots \\
&x_j = r.\sin(\theta_1)\dots\sin(\theta_{n-j})\dots\cos(\theta_{n-j+1}) \\
&\vdots \\
&x_n = r.\cos(\theta_1)
\end{aligned} \quad (3)$$

Where r ϵ [0, ∞], ϕ ϵ [0,2π] and θ ϵ [0,π]. The search space carried in n-dimensional space with (θ_1, θ_2, θ_{n-2}, ϕ, r).

D. Effects of Polar Coordinates Conversion

The transformations between Cartesian and polar search spaces have some effects. Matthysen and Engelbrecht found that the use of polar coordinates will generate a distorted version of search space compared with the Cartesian search space [5], making it more difficult to search in space (based on the kind of problem) to find out a global or local optimum [5]. Furthermore, the main effects of this distortion that may happen are that the particles may get stuck in a local optimum; the value of r may be brought near to the origin point, or the global optimum point may be lost by moving further away. Moreover, the distortion becomes greater when the number of dimensions increases. This distortion occurred during the execution of the algorithm by converting the position vectors from Cartesian to polar and vice versa by using equations 3 and 4, which are as follows:

$$\begin{aligned}
&\vec{\theta} = \eta(\vec{x}) \\
&r = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \\
&\theta_1 = \cos^{-1}(x_n / r) \\
&\theta_2 = \cos^{-1}(x_{n-1} / (r.\sin(\theta_1))) \\
&\vdots \\
&\theta_j = \cos^{-1}(x_{n-j+1} / (r.\sin(\theta_1)).\dots\sin(\theta_{j-1}))) \\
&\vdots \\
&\theta_{n-2} = \cos^{-1}(x_3 / (r.\sin(\theta_1)\dots\sin(\theta_{n-3})))
\end{aligned} \quad (4)$$

Where the value of the azimuth angle (ϕ) can derive by using the inverse equation as follows

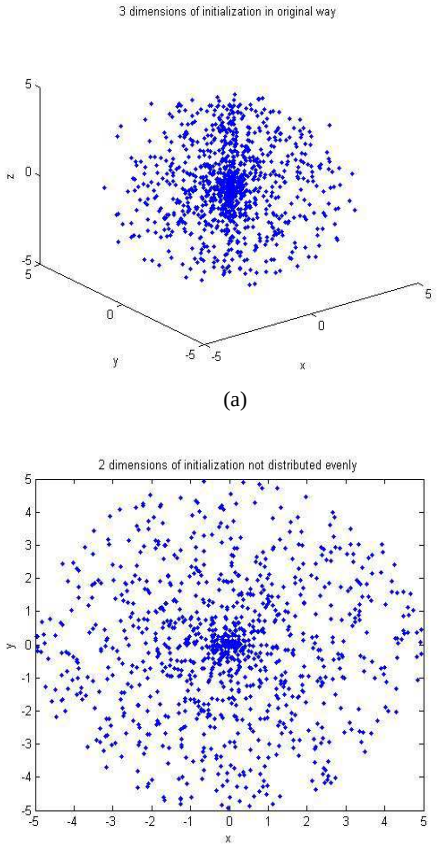
$$\phi = \cos^{-1}(x_1 / (r.\sin(\theta_1)...\sin(\theta_{n-2}))),$$
$$\phi = \sin^{-1}(x_1 / r.\sin(\theta_1)...\sin(\theta_{n-2}))) \quad (5)$$

To find out the value of the azimuth angle, both values of previous equations are used as well. Furthermore, the optimization algorithms that work in polar coordinates are suitable to solve special problems. Exclusively, polar coordinates offer a proper way to use a fixed value of r in a polar vector (r, θ_1, θ_2, θ_{n-2}, ϕ). The Evolution Strategy algorithm (ES) was modified to search in polar coordinates by solving the unit length projection vector [1]. The PES algorithm shows better results than its counterpart in Cartesian search space. In addition, to enable the polar ES to explore the search space in a convenient way, to produce better results it needs to narrow the search unit length vector in hypersphere that found in the search space by reducing it enough.

E. PSO Initialization in Polar Coordinates

PSO algorithm originally operated in Cartesian coordinates. However, [5] showed in their results that the use of polar coordinates initialization distorted search space, as was shown in their results. Their method was generated based on the general initialization equation, which can be presented as follows:

However, from their research, two and three dimensions initialization in polar coordinates were done and showed as follows in Fig. 1.



(a)

(b)

Fig. 1 Polar initialization diversities for 2d and 3d

The components were as follows: phi angle falls in [0, 2π] and distance falls in [0, R], where R represents the maximum distance between the origin and the point that can be the maximum distance from the origin. Moreover, [5] showed that the initialization of particle position in polar coordinates after the derivation of main initialization equation would be as follows:

*Initial phi = (2 * π - 0) * rand (0, 1) – 0*
*Initial phi = 2 * π * rand (0, 1)*

In addition, the distance initialization can be written as follows:

*Initial Distance = (R-0) * rand (0, 1) – 0*
*Initial Distance = R * rand (0, 1)*

Where the points will not be diverse enough based on the distance equations that were investigate based on the literature [5]. Random initialization of the particle positions in polar coordinates was not diverse enough in search space based on the distance r and the angle direction.

F. Proposed Initialization in Polar Coordinates

One of the main effects that caused a distortion during the use of polar coordinates in the search space is the initialization of particle positions. A new initialization technique will be introduced in this section. Furthermore, subsection E explained the initialization of the PSO algorithm in polar coordinates, showing that it was not uniformly distributed when converted back to Cartesian space. Two effects of this are that the positions of the particles gather to the origin point when the dimension increased, and the initial polar position is not diverse enough to cover the portion of search space. Moreover, distributing points uniformly in a sphere requires redefining the distance for all points based on the dimensionality of points. The distortion happened because the area of the element $d\Omega = (\sin\theta d\phi d\theta)$ which presents a function of azimuth angle θ and the points distributed randomly only as θ=π*random will generate points bunched to the poles and close to the origin. Furthermore, to avoid all matters that happened in the literature, use the Archimedes theorem [10] and Cumulative Densities Function (CDF). By inverting for the CDF, the distribution of the zenith and azimuth angles will be:

$$\phi = 2\pi rand(0,1) \quad (6)$$
$$\theta = \cos^{-1}(2rand(0,1) - 1) \quad (7)$$

In addition, generating uniformly random number distributions in a true way within a circle can be done by using dynamic radius R in the (x, y, z, …, n) plane. At initial polar coordinates, this seems a great idea, and the simple solution for this idea is to pick an inner radius r uniformly within the range [0, R]. Moreover, to avoid the points being distributed near the origin (0, 0), there are more points that need to be generated further out (at large value of r) the radius must generate by following systematic distribution not only in a uniform way. To do this in a proper way, one must do as follows:

$$r = R \cdot rand(0,1)^{(1/\dim)} \quad (8)$$

Fig. 3 presents the plotting of new proposed method by using the randomly distributed generator for two and three dimensions as follows:

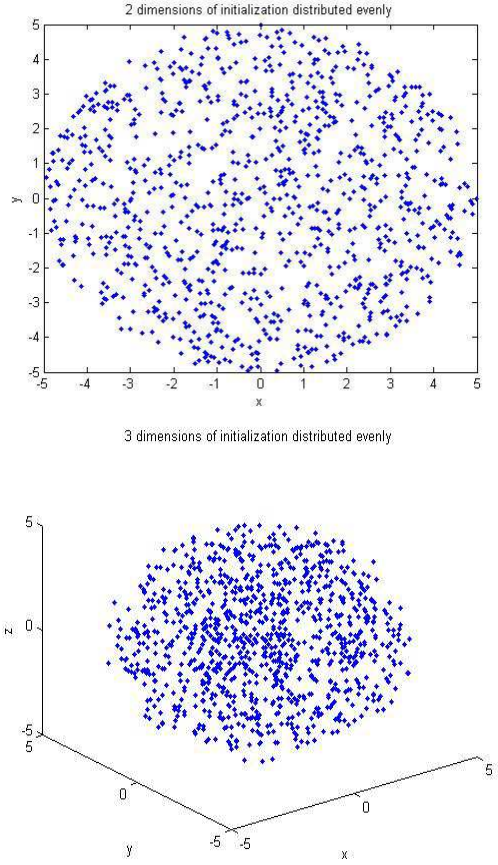


Fig. 3 Initial 1000 points in two and three dimensions uniformly

1) *Normal Truncated Initialization Distribution in Polar PSO*

To further investigate the introduced polar initialization in polar coordinates, the use of various numbers of other initialization distributions will give a good method to test our assumptions. Examples of randomization within the range [0, 1] are Erlang distribution, Beta distribution, Arcsine distribution, Rayleigh distribution, inverse Gaussian distribution and truncated normal distribution. Furthermore, by testing the previous type we found that the truncated normal distribution has in common with uniform distribution the parameters that are used in it : a- lower truncation point; b- upper truncation point; (b>=a); y - mean of the parent normal distribution and standard deviation of the parent normal distribution (sigma>0) where a <= y <= b. where Fig. 4 presents the plotting of 2d and 3d for truncated initialization to show the diversity of initial points based on the proposed method.

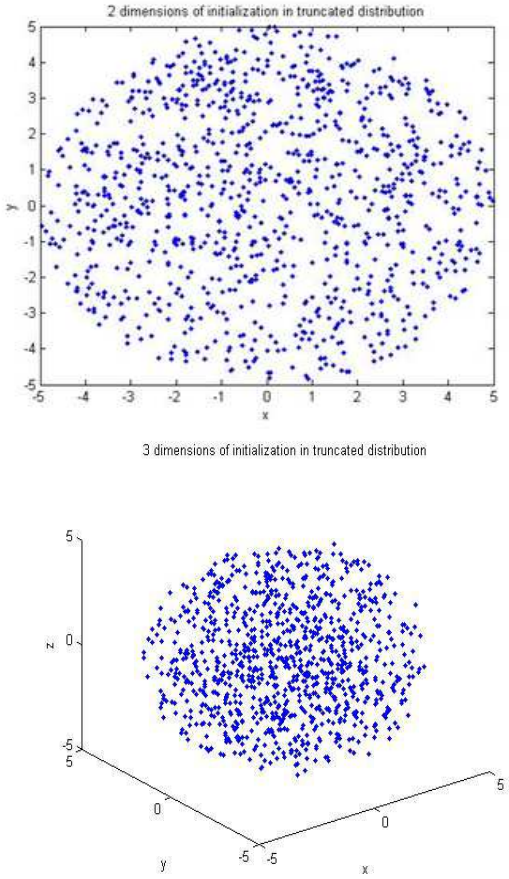


Fig. 4 normal truncated initialization for 1000 points in two and three dimensions

2) *Possible Scenarios of Initialization*

To cover the search space corners by using the polar coordinates initialization introduced in section III-A, the value of maximum distance R should stretch to discover the corners of search space. The same scenario will apply the proposed method by modifying the value of R. Moreover, use of the random initialization distribution in polar coordinates has an effect; where the search space becomes enlarged and needs to enforce the particle to be in the valid range. However, the best way is by restricting the particle to re-initialize its position by following the same zenith and azimuth angle distribution as mentioned in subsection III-A. The redefined equations for the inner distance in the sphere will be modified to find the appropriate distance to be within the range of benchmark problems, which are presented as follows:

$$r = Ubound \cdot rand(0,1)^{(1/\dim)} \quad (9)$$

Fig. 5 presents the constraint that used to be reinitialized for the points generated out of the boundary to be in boundary range, by redefining the maximum value of distance R to be less or equal to the maximum value. As an example, in DeJong benchmark test function the boundary of search space is [-5.12, 5.12] where the value of R will be 5.12, and the plotting will be as follows:

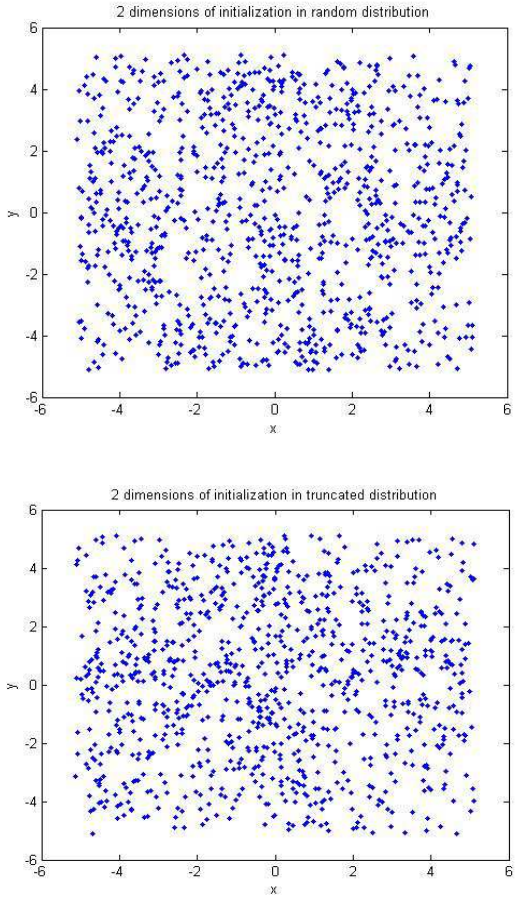


Fig. 5 illustrates the Random Distribution initialization in polar space with the constraint to enforce the initial position handle in the search space in a randomly distributed way and truncated normal distribution.

III. Results and Discussion

This section explains the experimental results with its settings while comparing the performance of standard PSO algorithm operated in Cartesian coordinates and polar coordinates. Furthermore, the proposed new initialization technique in polar coordinates will be investigated by using the PSO algorithm as a well-known algorithm. Moreover, an enhancement of polar PSO using the proposed technique with scenarios of initialization distribution copies will be described. The parameter values were selected based on the guidelines of [8] for whenever c1 and c2 are equal to 1.496180 and w being set to 0.729844. All of these values were adjusted to be suitable for the condition that allows the algorithm for convergent particle trajectories with the guidelines defined in [8]. A collection of 20 particles was chosen to run the versions of algorithms. In addition, 1000 maximum iterations were run 100 times for all the versions of algorithms in sequence, printing the generated results to Excel files without modification. All results were averaged and reported by following ANOVA one-way test as in [11], which will be summarized in the discussion subsection. Furthermore, there are many benchmark testing functions that were proposed to check the performance of optimization algorithms [12], [13]. In addition, to develop and investigate any enhancement or modify any optimization algorithm, it must first be validated by using these benchmark testing functions, as well as some shifting should occur to enforce the search of particles avoiding the origin point. This shifting is illustrated in Table 1. Table 2 illustrates the versions of both algorithms' copies.

TABLE I
Benchmark Test Functions

Function name	Function equation	Shifting	Domain
Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}(x_i - shift)^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi(x - shift))\right) + 20 - e$	-10	[-32, 32]
DeJong	$f(X) = \sum_{i=1}^{d}(x_i - shift)^2,$	-2.5	[-5.12, 5.12]
Rastrigin	$f(x) = 10d + \sum_{i=1}^{d}[(x_i - shift)^2 - 10\cos(2\pi x_i - shift)],$	-2.5	[-5.12, 5.12]
Rosenbrock	$f(x) = \sum_{i=1}^{d-1}[100((x_{i+1} - shift) - (x_i - shift)^2)^2 + ((x_i - shift) - 1)^2],$	0.0	[-2.048, 2.048]
Griewangk	$f(x) = \sum_{i=1}^{D}\frac{(x_i - shift)^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{(x_i - shift)}{\sqrt{i}}\right) + 1$	-300.0	[-600, 600]
Quartic	$f(x) = \sum_{i=1}^{D} i(x_i - shift)^4$	-0.5	[-1.28, 1.28]
Salomon	$f(x) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{d}(x_i - shift)^2}\right) + 0.1\sqrt{\sum_{i=1}^{d}(x_i - shift)^2}$	-300.0	[-600, 600]
Dixon	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 + x_{i-1})^2$	0.0	[-10, 10]

TABLE II
DESCRIBED THE SETTING OF PSO GENERATIONS FOR ALL BENCHMARK TEST FUNCTIONS

Version of modifications	**Description**
V1	Standard PSO with Cartesian initialization
V2	Standard PSO with polar initialization standard technique (undistributed)
V3	Standard PSO with polar initialization **modified technique** (distributed)
V4	Original polar PSO with cartesian initialization
V5	Original polar PSO with polar initialization using **modified technique** (distributed)
V6	Original polar PSO with polar initialization using **modified technique truncated distribution** (distributed)
V7	Original polar PSO with polar initialization using **modified technique with maxDistance without constraint**
V8	Original polar PSO with polar initialization using **modified technique with maxDistance with constraint**
V9	Original polar PSO with polar initialization using **modified technique with maxDistance truncated distributed with constraint.**

A. PSO Results

In this section a comprehensive ANOVA to compare the mean of three or more algorithms provided subsection by analysing the sample variances [11]. Table 3 shows the mean and standard deviation of DeJong benchmark test function for all settings of PSO with its initialized versions.

TABLE III
THE PERFORMANCE RESULTS FOR DEJONG TEST FUNCTION

Version of modification	**Dimensions**						
	3	5	10	20	30	50	100
V1	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00	1.08e-19 ± 8.52e-19	1.52e-07 ± 1.05e-06	5.62e-02 ± 2.21e-01	1.27e+01 ± 9.11e+00
V2	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00	3.38e-20 ± 2.64e-19	1.21e-08 ± 8.33e-08	1.71e-02 ± 8.21e-02	4.36e+00 ± 2.77e+00
V3	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00	2.37e-21 ± 7.45e-21	6.04e-10 ± 1.91e-09	6.88e-03 ± 1.31e-02	4.21e+00 ± 2.31e+00

In Table 3, for the first of three settings (3, 5, 10 dimensions) the same results come out where all versions of the algorithm reach the optimal point. In 20, 30 dimensions the results show that the modified algorithm by using the original method of initialization was enhanced a bit compared with the original PSO. However, the V3, which presents the use of the new proposed distributed method to make an initialization in polar coordinates, showed better results. All in all, the experimental results show some differences in mean and standard deviation in some cases.

Statistical proved that there is no significant difference. In Table 4 the significant differences showed that there are no significant differences between the original PSO and its modified versions. Statistically, it is not significant. The same result occurred with all benchmark test functions that used such as Ackley, Rastrigin, Rosenbrock, Griewangk, Quartic, Salomon and Dixon.

TABLE IV
SHOW THE SIGNIFICANT DIFFERENCES BETWEEN THE ORIGINAL PSO WITH V2 AND V3

Benchmark function	Ackley	Dejong	Dixon	Griewangk	Rastrigin	Quartic	Rosenbrock	Salomon
P_VALUE	0.825666	0.700643	0.393024	0.790116	0.708982	0.484251	0.47522	0.916853

B. Polar PSO Results

In this subsection, many versions of polar PSO were done with many types of initialization scenarios. Furthermore, the experimental results were statistically compared by using ANOVA. Table 5 show the mean and standard deviation of Dejong benchmark test function for all versions of polar PSO algorithm, from V4 to V9. Various benchmark test functions were used in evaluating all modified versions of polar PSO such as Ackley, Rastrigin, Rosenbrock,

Griewangk, Quartic, Salomon and Dixon. Table 6 shows ANOVA for all benchmark test functions used to evaluate the modified versions of polar PSO.

TABLE V
THE PERFORMANCE RESULTS FOR DEJONG TEST FUNCTION

Polar PSO versions	**Dimensions**						
	3	5	10	20	30	50	100
V4	1.00e-01 ± 4.51e-01	1.28e-01 ± 5.02e-01	5.10e-01 ± 1.12e+00	6.83e-01 ± 8.40e-01	3.60e+00 ± 5.92e+00	3.40e+01 ± 2.66e+01	2.29e+02 ± 7.33e+01
V5	3.71e-01 ± 1.06e+00	4.85e-01 ± 1.02e+00	6.96e-01 ± 1.37e+00	1.62e+00 ± 4.11e+00	6.71e+00 ± 1.16e+01	7.60e+01 ± 4.68e+01	3.92e+02 ± 5.71e+01
V6	3.86e-01 ± 1.07e+00	5.62e-01 ± 1.36e+00	5.16e-01 ± 1.47e+00	1.22e+00 ± 1.67e+00	9.95e+00 ± 1.74e+01	7.72e+01 ± 4.11e+01	3.81e+02 ± 6.03e+01
V7	3.99e-01 ± 7.84e-01	7.18e-01 ± 1.18e+00	7.56e-01 ± 1.49e+00	1.38e+00 ± 2.06e+00	9.05e+00 ± 1.22e+01	7.85e+01 ± 4.92e+01	3.88e+02 ± 4.71e+01
V8	4.29e-01 ± 1.45e+00	5.69e-01 ± 1.31e+00	7.10e-01 ± 1.14e+00	1.50e+00 ± 4.06e+00	9.20e+00 ± 1.60e+01	6.25e+01 ± 4.49e+01	3.47e+02 ± 5.66e+01
V9	2.45e-01 ± 6.32e-01	4.11e-01 ± 9.18e-01	4.72e-01 ± 7.99e-01	1.16e+00 ± 1.68e+00	7.54e+00 ± 1.35e+01	5.98e+01 ± 3.83e+01	3.56e+02 ± 5.52e+01

TABLE VI
POLAR PSO ANOVA STATISTICAL TEST

Benchmark function	Ackley	Dejong	Dixon	Griewangk	Rastrigin	Quartic	Rosenbrock	Salomon
P_VALUE	0.999604	0.997998	0.052902	0.998377	0.654213	0.991411	0.992975	0.998781

IV. CONCLUSION

This paper investigated the polar PSO by re-initializing the particle positions based on the proposed initialization technique in polar coordinates, where it is distributed more evenly than the normal method of polar initialization used in previous research. The significance of using a new method of initialization with polar PSO achieves greater success by removing the distortion that occurred. Experimental results showed that the initialization did not affect the PSO movement to get out of being stuck in local optima. The previous research mentioned that the problem is caused by the difficulty of floating points because of the transformation from polar to Cartesian coordinates. Furthermore, various types of modification were done in this research for both PSO and Polar PSO. The significance of using PSO in research is that the PSO algorithm is known as a stable algorithm. The comparison was based on applying all experimental approaches for all settings of modified versions. Based on the results the proposed initialization technique showed little enhancement in PSO algorithm by comparing all versions of modifications. Some of the benchmark functions outperform the original copy of PSO by a small percent, but statistically, there are no significant differences. From the results of polar PSO algorithm and PSO algorithm, we can notice that the problem happened because of the logical mapping velocity equation of PSO algorithm stuck in local optima. By returning to the concept of polar coordinates that zenith and azimuth angles should update by returning to the origin point which is not used in the original polar PSO, future directions need to modify the velocity equation of PSO to allow the algorithm to search in polar space or introduce a new algorithm in polar coordinates based on the natural behaviour of specific types of particles.

ACKNOWLEDGEMENT

This work was supported by the Department of Higher Education of Malaysia under Grant FRGS/1/2016/ICT02/UKM/02/1.

REFERENCES

[1] A. Sierra, A. Echeverría, The polar evolution strategy, in: Proceedings of the IEEE Congress on Evolutionary Computation, July 2006, pp. 2301–2306.

[2] J. Kennedy, R.C. Eberart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, November 1995, pp. 1942–1948.

[3] R. Storn and K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (December (4)) (1997) 341–359.

[4] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of the IEEE International Conference on Computational Cybernetics and Simulation, vol. 5, October 1997, pp. 4104–4108.

[5] W. Matthysen , A.P. Engelbrecht, A polar coordinate particle swarm optimizer, applied soft computing journal. 2011, pp.1322-1339.

[6] M. Shatnawi, M.F. Nasrudin and S. Sahran, Starting Configuration of the Polar Particle Swarm Optimizer Algorithm in Polar Coordinates. Proceeding of CAITPS (2015), 59-64.

[7] A. D. Bambang, Y. Hendrawan, D. F. Al-Riza and A. N. J. Laksono, Optimization of PID Controller Parameters on Flow Rate Control System Using Multiple Effect Evaporator Particle Swarm Optimization. IJASEIT, vol 5. No 2. 2015.

[8] F. van den Bergh, An analysis of particle swarm optimizers, Ph.D. Dissertation, Department of Computer Science, University of Pretoria, South Africa, 2002.

[9] M.G. Kendall, a Course in the Geometry of n Dimensions, Dover Publications, 2004.
[10] Cundy, H. and Rollett, A. "Sphere and Cylinder--Archimedes' Theorem." 4.3.4 in Mathematical Models, 3rd ed. Stradbroke. England: Tarquin Pub., pp.172-173, 1989.
[11] Mario F. T., “elementary statistics”, Eighth Edition. Copyright 2001. Addison Wesley Longman.
[12] P. N. Suganthan, N. Hansen , J. J. Liang, K. Deb, Y. -P. Chen, A. Auger, S. Tiwari.,” Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization”. 2005, Technical Report, Nanyang Technological University, Singapore.
[13] Molga and Smutnicki 2005. Test functions for optimization needs.