International Journal of Soft Computing 12 (4): 228-235, 2017
ISSN: 1816-9503

Particles Initialization of the Polar Particle Swarm Optimizer (Polar PSO) Algorithm in Polar Coordinates

Moaath Shatnawi, Mohammad Faidzul Nasrudin and Shahnorbanun Sahran
Faculty of Information Science and Technology, Center for Artificial Intelligence Technology (CAIT),
University Kebangsaan Malaysia, Bangi, Selangor, Malaysia

Abstract: Polar Particle Swarm Optimizer (Polar PSO) is a modified version of Particle Swarm Optimization (PSO) algorithm that used a mapping function that takes position of particles in polar space and converts them to Cartesian space and vice versa. The conversion is necessary since the particles are initialized and evaluated in Cartesian space while their movements are in polar space. The conversion however distorts the position of particles even though they were initially uniformly distributed. So, this conversion is believed to be the reason behind the Polar PSO performs poorly compared to the original Cartesian PSO, especially in high dimensions. This study proposes an initialization method in polar space for Polar PSO. It uses a distribution function to avoid the points being distributed near the polar origin. This method will reduce the number of conversion and in the same time diverse the position of particles to cover a sufficiently large portion of the search space. The proposed method is tested in Ackley, DeJong, Rastrigin, Rosenbrock, Griewangk, Quartic, Salomon and Dixon benchmark functions. The results show that the polar initialization improves slightly the performance of the Polar PSO. Although, the polar initialization is useful in reducing the distortion during conversion the Polar PSO can be further improved by enhancing its movement in polar space.

Key words: Polar coordinates, Polar Particle Swarm Optimization, random polar initialization, uniformly distributed, polar space

INTRODUCTION

Stochastic optimization algorithms such as the PSO were inspired based on the flocking and shoaling behaviour of birds and fish which introduced by Kennedy and Eberhart (1995) to present the simulation of this behaviour as a new model of computational intelligence field. It was designed as a simple, easy and efficient algorithm to solve several n dimensional benchmark optimization problems where operates in Cartesian space. Many modified versions of the algorithm had emerged to solve various problems in diverse areas. The initial algorithm has been modified by improving its fundamental logic or using another appropriate mapping function (Bergh, 2002).

An example of fundamental logic modification in PSO algorithm is the discrete binary PSO, a modification to make the algorithm able to operate on discrete binary variables by searching for discrete solutions in binary space (Kennedy and Eberhart, 1997). It uses each component of particles velocity vectors to find out the probability where a coordinate will take on a zero or one.

On the other hand, the Angle Modulated PSO (AMPSO) is an example of mapping function utilization into PSO algorithm (Pampara *et al*., 2005). It was developed mainly to reduce the complexity of binary problems by evolving the coefficient function of a trigonometric model in four dimensional Euclidean spaces. This technique was used with the differential evolution algorithm to reduce the complexity in discrete problems (Pampara *et al*., 2006).

Another, example of exploitation of mapping function in PSO is in the Polar PSO. The original PSO is redefined by using an appropriate mapping function to search in polar coordinates by using the transformation technique from Cartesian to polar and vice versa (Matthysen and Engelbrecht, 2011). However, experimental results show that the Polar PSO had an inferior results comparing with its counterpart in Cartesian space. This might be due to the bad particles initialization caused by the coordinate transformation from polar to Cartesian. According to Matthysen and Engelbrecht (2011) Polar PSO shows some complications such that the new search space had become a distorted version of original search space (Kendall, 2004).

Corresponding Author: Moaath Shatnawi, Faculty of Information Science and Technology,
Center for Artificial Intelligence Technology (CAIT), University Kebangsaan Malaysia, Bangi, Selangor, Malaysia

MATERIALS AND METHODS

Particle Swarm Optimization (PSO): PSO algorithm is categorized as a population based algorithm which contains the swarm of particles which are distributed in n-dimensional space. Each particle position represents the possibility solution for optimization problem by finding the fitness value of each particle where it is used to find out the personal best position. A local best position represents the best position of the current particle in each execution of the algorithm. It is also a historical record of personal best position of each particle in order to find the best value of fitness which represents the global best position of the best particle (Bergh, 2002). Through the algorithm two kinds of variant topologies have been defined which are the lbest (local best) and gbest (global best). Both of them are used to help the particles to search more effectively in the search space by discover promising regions that will show the way to further exploration. All of this information is shared through the rest of particles. Let n be the size of population (swarm). Each particle i represented with varies characteristics were show as follows:

- x_i: The current position of the particle
- v_i: The current velocity of the particle
- y_i: the personal best position of the particle

Based on the above characteristics of particle i, the velocity of each particle is calculated as follows by using this rule:

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

For all dimensions where w is the interia weight which define the effect of its previous velocity on the current velocity; $\alpha_1(t)$ and $\alpha_2(t)$ are defined as $\alpha_1(t) = c_1 \times r_1(t)$ and $\alpha_2(t) = c_2 \times r_2(t)$ where $r_1(t)$ and $r_2(t)$ are a random values (0, 1) and c_1 and c_2 are the acceleration constants to determine the effect of personal best position $y_{i,j}(t)$ and global best position $\hat{y}_{i,j}(t)$ on the new velocity value. The new position of the particle will be updated by using the previous equation and the current position of the particle. The update equation presents as follows:

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

The new position is taken based on the position of global and local best positions as a vector.

Polar particle swarm optimizer: Polar particle swarm optimizer is one of the latest modified versions of PSO which is proposed by Matthysen and Engelbrecht (2011). PSO is originally operated by producing solution vectors in Cartesian space. However, Polar PSO is operated in polar search space by using an appropriate mapping function and producing polar solution vectors. Its algorithm converts position vectors from Cartesian coordinates to polar coordinates and vice versa to calculate fitness values accordingly.

The Polar PSO algorithm relies on the polar conversion function to convert a position from Cartesian to polar and vice versa. It also uses a bounded angular function to limit the particles in the defined region. Both of the functions are describe in the following sub sections.

Polar conversion function: To allow a Polar-Based algorithm to evaluate fitness functions in Cartesian space, it requires a mapping conversion function in n-dimensional from polar vector to Cartesian. It is formulated by Kendall (2004) as follows:

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

where, $r \in [0, 8]$, $\phi \in [0, 2\pi]$ and $\theta \in [0, \pi]$. The search space is carried in n-dimensional space with polar dimensions $\theta_1, \theta_2, \ldots, \theta_{n-2}, \phi_r$ and their respective Cartesian dimensions $x_1, x_2, \ldots, x_{n-2}, x_{n-1}, x_n$.

Effects of polar coordinates conversion: The transformations between polar and Cartesian search spaces have some effects. The main effect is the search space become a distorted version of search space. According to Matthysen and Engelbrecht (2011), the use of polar coordinates will generate a distorted version of search space comparing with the Cartesian search space.

The consequences of this distortion is that particles may get stuck in local optimum because of the value of r is near to the origin point or the global optimum point lose by move further away. On other hand, the transformation from Cartesian to polar coordinates effects the value of distance r while the number of dimensions is increased. Which lead to stretch the search space which makes enlargement of local optima position and lose the global optima as well:

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

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

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

To find out the value of azimuth angle both values of previous equation used as well. Based on Matthysen and Engelbrecht (2011), the polar coordinates can be presents by the form (r, θ_1, θ_2, ..., θ_{n-2}, ϕ) for each particle polar vector where $0 \le r \le \infty$, $0 \le \theta_{n-2} \le \pi$, $0 \le \phi \le 2\pi$ were used in the particle positions velocity update rules. Furthermore, these angles need to be within the range to keep the particles in defines region by using (Eq. 1 and 2). Add to that, the effects of these constraints angular will effect by reducing the size of search space to allow the PSO algorithm explore the search space well.

Two kinds of boundary arrange will redefine. The first belong to the azimuth angle (ϕ) which update during the velocity update step by using the modular arithmetic were used to keep the azimuth angle (ϕ) in define area. These updating rules can presents as follows:

$$x_{i,j} = x_{i,j}(t-1) + v_{i,j}(t) \tag{6}$$

For dimension j = 1, ... , d-1 and:

$$x_{i,d} = \begin{cases} (x_{i,d}(t-1) + v_{i,d}(t)) \bmod 2\pi + 2\pi \\ \cdots \text{if} x_{i,d}(t-1) + v_{i,d}(t) < 0, (x_{i,d}(t-1) \\ + v_{i,d}(t)) \bmod 2\pi \cdots \text{otherwise} \end{cases} \tag{7}$$

This updated rule seems a periodic rule as described in Matthysen and Engelbrecht (2011). Add to that, the negative values of polar angle will handle by adding 2π to find out the equivalent positive angle. Ignoring this rule will carry out inefficient results.

Regarding the azimuth angle (ϕ) behavior in mathematics which restricted in range (0, 2π) the value may be become out of boundary during velocity update rules. In this regards, the velocity update rules should modify to handle without any problem by following this new updating rule:

$$\begin{aligned} v_{i,j}(t) = {} & wv_{i,j}(t-1) + \alpha_1(t)(y_{i,j}(t) - \\ & x_{i,j}(t)) + \alpha_2(t)(\hat{y}_{i,j}(t) - x_{i,j}(t)) \end{aligned} \tag{7}$$

For dimension j = 1, ... , d-1 and:

$$v_{id} = wv_{i,d}(t-1) + \begin{cases} \alpha_1(t)(y_{i,d}(t) - x_{i,d}(t)) \cdots \\ \text{if}(y_{i,d}(t) - x_{i,d}(t)) \le \pi \\ \alpha_1(t)(\text{sign}(x_{i,d}(t) - y_{i,d}(t)) \times 2\pi - x_{i,d} \\ (t) + y_{i,d}(t)) \cdots \text{otherwise} \end{cases} + \begin{cases} \alpha_2(t)(\hat{y}_{i,d}(t) - x_{i,d}(t)) \cdots \\ \text{if}(\hat{y}_{i,d}(t) - x_{i,d}(t)) \le \pi \\ \alpha_2(t)(\text{sign}(x_{i,d}(t) - \hat{y}_{i,d}(t)) \times 2\pi - x_{i,d} \\ (t) + \hat{y}_{i,d}(t)) \cdots \text{otherwise} \end{cases} \tag{8}$$

The second part of these angles is the polar angle (θ) which refer to the angles (θ_1, θ_2, ..., θ_{n-2}) of the polar particle position vector. Which are restricted in mathematical behavior within the range (0, π). Furthermore, update these angles in similar way of azimuth angle will led to get a large step of movement even if polar angle move in small amount.

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 were the diversities in polar and Cartesian coordinates are differ. The conversion from Cartesian to polar and vice versa in term of initialization face problems were explained in the next section.

Polar initialization in PSO algorithm: Random initialization the particles position in polar coordinates does not divers enough in search space based on the distance r and the angle direction. Comparing with initialization in Cartesian space were covered the search space. The matter happened when make a transformation between the Cartesian and polar coordinates in terms of initialization were the diversity will be differ. Based on (Matthysen and Engelbrecht, 2011) the authors showed the differences between both of initialization ways.

Figure 1 shows the different between both the transformation of Cartesian and polar coordinates diversities in search space in term of initialization. Figure 1a shows the positions of particles that were initialized originally in polar coordinates by initialize the distance and polar angles with azimuth angle then convert the positions from polar to Cartesian. Whilst, Fig. 1b shows the diversity of particle positions were initialized in term of Cartesian space then convert them to polar coordinates.

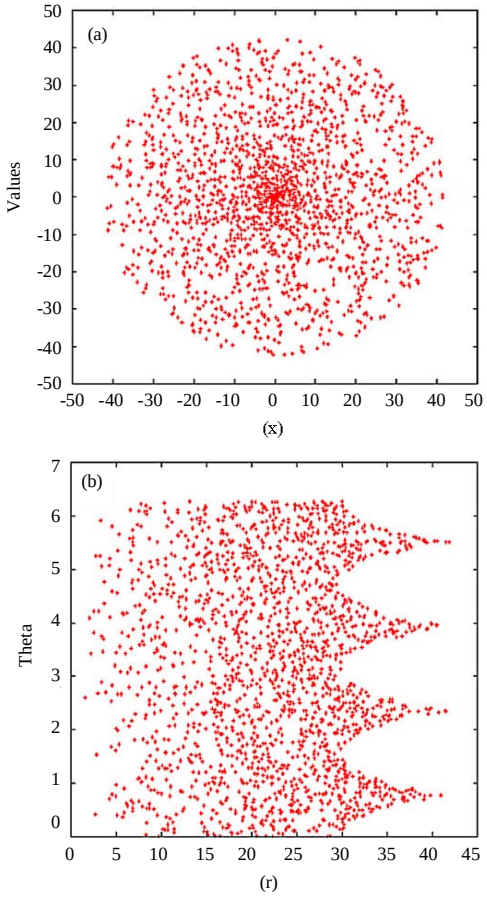

Fig. 1: a, b) Conversion of particle position from polar to Cartesian

Polar initialization technique in polar coordinates for Polar PSO: The Polar PSO algorithm initialized in polar coordinate space were not uniformly distributed when converted back to Cartesian space. Tow effects in this side where represents that the initial polar position not diverse enough to cover the portion of search space and the position of particles gather to origin point when the dimension is increase.

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 $\theta = \pi^*$ random will generate points bunched to the poles and close to origin (Fig. 2).

Furthermore, to avoid all matters that happened in the literature, use the Archimedes theorem (Cundy and Rollett, 1989) and Cumulative Distribution Function (CDF). By inverting the CDF the distribution of the zenith and azimuth angles will be:

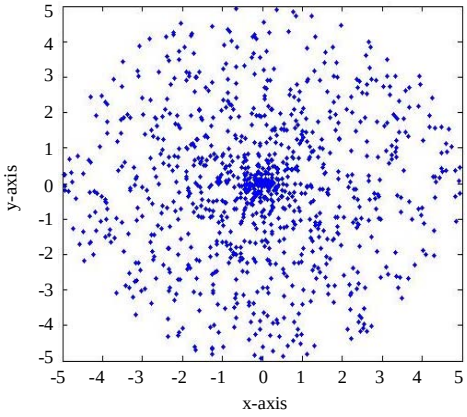

Fig. 2: The distribution of particles when r and θ are randomized: 2 dimensions of initialization not distributed evenly

$$\phi = 2\pi\,\text{rand}(0,1) \tag{9}$$

$$\theta = \cos^{-1}(2\text{rand}(0,1) - 1) \tag{10}$$

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 follow:

$$r = R \times \text{rand}(0,1)^{(1/\dim)} \tag{11}$$

Where the plotting of previous equations are shown as the initialized particle positions using the proposed PI technique shows some particle positions drawn outside of boundaries. An example in Fig. 3, the maximum boundary should be 5.12 for dejong test function. This matter occurred cause of the value of maximum distance (R). In order to solve this matter re-initialization shall be occurs by restricting the particle to re-initialize its position by following the same zenith and azimuth angle distribution by following Eq. 9 and 10. 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:

$$r = \text{Ubound} \times \text{rand}(0,1)^{(1/\dim)} \tag{12}$$

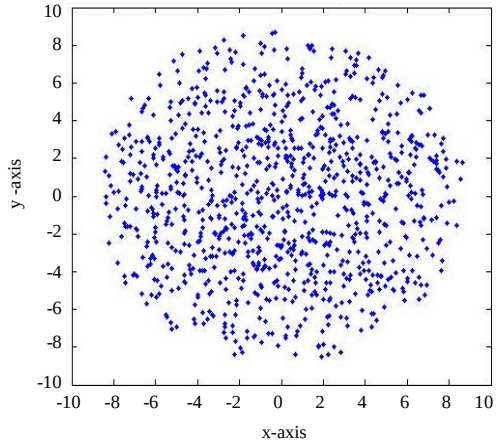

Fig. 3: The distribution of particles using the initial proposed method: 2 dimensions of initialization evenly distributed

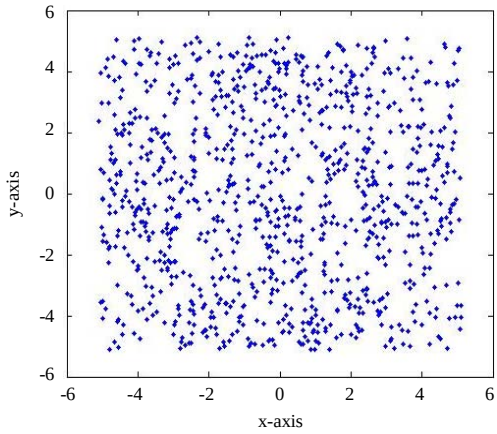

Fig. 4: The distribution of particles using the proposed method with constraint: 2 dimensions of initialization in random distributed

where, the Ubound is the maximum boundary of benchmark test function. The pseoudo code of PI technique is as follows:

Algorithm 1 (Initialization technique in polar coordinates with constraints):

```
For each particle i Do
    Initialize theta angle using Eq. (9)
    Initialize phi angle using Eq. (10)
    Find the max distance (R)
    Set R value by using Eq. (11)
//start constraint
    If (the particle position out of boundary)
        Re-initialize particle position using Eq. (12)
    End
//end constraint
End
```

Table 1: Settings of algorithms

Setting number	Setting description	Initialization space
S1	Standard PSO	Cartesian
S2	Polar PSO	Cartesian
S3	Polar PSO	Polar

Table 2: Benchmark test functions

Function	Shifting	Domain
Ackley	-10.0	[-32.768, 32.768]
DeJong	0-2.5	[-5.12, 5.12]
Rastrigin	0-2.5	[-5.12, 5.12]
Rosenbrock	000.0	[-2.048, 2.048]
Griewank	-300.0	[-600, 600]
Quartic	00-0.5	[-1.28, 1.28]
Salomon	-300.0	[-600, 600]
Dixon	000.0	[-10, 10]

Figure 4 presents the constraints that used to re-initialize the points generated out of the boundary to be in valid boundary region using random distribution.

Experimental settings: The proposed polar initialization method is tested on several benchmark test functions. Then, it is compared with those results from the Cartesian-Initialized PSO and Cartesian-Initialized Polar PSO. For the purpose of referencing, accordingly the first algorithm is set as S1 the second algorithm as S2 and the proposed algorithm is set as S3. The settings are listed in Table 1. To further complicate the solution of the fitness functions, shifting of value is used. Solution for each fitness function is shifted to a certain value as listed in Table 2.

For the standard PSO, the parameter values were selected based on the guidelines of (Bergh, 2002) where both c1 and c2 are equal to 1.496180 and w is being set to 0.729844. A collection of 40 particles is chosen for all experiments. Each fitness function is run 100 times with a maximum of 1000 iterations. Error rate with its standard deviation for each fitness function is logged into an excel files for reporting. All experiments are computed in a computer with Intel Core i7 with 8 GB RAM.

RESULTS

Comparison results of initialized versions of Polar PSO with the Cartesian PSO and standard version of Polar PSO are mentioned and discussed in this study. About 3.1 provide the analysis in details and discussion of the Rosenbrock benchmark test function. The results in other benchmark functions are provided in study.

Rosenbrock benchmark function: Table 3 shows the Rosenbrock test function for all settings of original PSO and Polar PSO with its three types of modified versions. The results show the varies of differences between the modified versions results comparing with Polar PSO in order of dimension size in 5 dimensions all modified

Table 3: The performance results for Rosenbrock test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	4.53E-06	4.97E-02	7.69E-01	1.13E+01	2.69e+01	5.81e+01	2.56e+02
	(6.67E-06)	(3.93E-01)	(1.04E+00)	(6.56e+00)	(1.65e+01)	(2.45e+01)	(8.34e+01)
2	7.65e-01	3.55e+00	8.40e+00	1.87e+01	2.87e+01	4.85e+01	9.84e+01
	(1.50e+00)	(2.91e+00)	(1.15e+00)	(4.86e-02)	(3.06e-02)	(5.17e-02)	(9.78e-02)
3	2.89e+00	3.49e+00	5.17e+00	7.84e+00	1.73e+01	4.86e+01	9.85e+01
	(5.22e+00)	(3.81e+00)	(3.97e+00)	(3.04e+00)	(1.13e+01)	(7.00e-02)	(1.10e-01)

Table 4: The performance results for Quartic test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	0.00E+00	0.00E+00	0.00E+00	7.27E-47	2.58E-28	1.15E-11	3.74E-02
	(0.00E+00)	(0.00E+00)	(0.00E+00)	(3.26E-46)	(1.10E-27)	(8.08E-11)	(1.14E-01)
2	1.10E-04	7.28E-05	6.89E-05	6.22E-04	4.80E-03	1.94E-01	2.10E+01
	(8.93E-04)	(4.17E-04)	(2.43E-04)	(4.56E-03)	(2.74E-02)	(5.58E-01)	(1.49E+01)
3	7.52E-05	3.09E-04	1.23E-04	1.65E-03	2.09E-02	1.71E+00	8.87E+01
	(3.34E-04)	(2.24E-03)	(4.80E-04)	(7.14E-03)	(4.72E-02)	(2.59E+00)	(3.11E+01)

Table 5: The performance results for Dixon test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	3.70e-32	1.67e-01	5.60e-01	6.60e-01	6.78e-01	4.21e+00	5.34e+02
	(0.00e+00)	(2.90e-01)	(2.46e-01)	(6.67e-02)	(7.68e-02)	(3.40e+00)	(4.94e+02)
2	2.51e-01	5.29e-01	6.68e-01	6.87e-01	7.44e-01	9.55e-01	1.00e+00
	(4.44e-01)	(2.87e-01)	(1.67e-03)	(5.30e-02)	(9.86e-02)	(5.33e-02)	(8.10e-04)
3	2.48e-01	5.10e-01	6.72e-01	7.32e-01	8.67e-01	9.98e-01	1.00e+00
	(4.30e-01)	(2.80e-01)	(2.35e-02)	(9.85e-02)	(1.19e-01)	(8.98e-03)	(0.00e+00)

Table 6: The performance results for Ackley test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	8.88e-16	9.06e-16	2.33e-15	5.03e-01	1.56e+00	4.18e+00	1.51e+1
	(0.00e+00)	(1.78e-16)	(8.26e-16)	(7.31e-01)	(9.63e-01)	(1.50e+00)	(2.40e+00)
2	3.29e-01	1.13e+00	2.43e+00	1.20e+01	1.62e+01	1.71e+01	1.73e+01
	(1.30e+00)	(1.93e+00)	(1.72e+00)	(5.88e+00)	(2.08e+00)	(3.18e-01)	(3.76e-02)
3	2.22e+00	2.11e+00	5.19e+00	1.67e+01	1.70e+01	1.71e+01	1.72e+01
	(2.69e+00)	(2.00e+00)	(4.94e+00)	(7.39e-01)	(1.35e-01)	(7.39e-02)	(3.26e-02)

Table 7: The performance results for Dejong test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	0.00E+00	0.00E+00	0.00E+00	1.11e-28	1.56e-16	2.45e-05	2.06e+00
	(0.00E+00)	(0.00E+00)	(0.00E+00)	(2.92E-28)	(1.04E-15)	(1.93E-04)	(2.84E+00)
2	5.32e-03	2.71e-02	6.48e-02	1.95e-01	5.36e-01	8.28e+00	1.88e+02
	(2.78E-02)	(1.18E-01)	(1.22E-01)	(2.30E-01)	(5.44E-01)	(1.19E+01)	(5.95E+01)
3	6.20e-02	1.44e-01	2.14e-01	2.26e-01	1.39e+00	2.59e+01	3.16e+02
	(2.01e-01)	(2.62e-01)	(4.05e-01)	(2.10e-01)	(3.76e+00)	(2.74e+01)	(6.34e+01)

versions of S2 show better results where similar performance for 10 dimensions happened. In 20 dimensions S3 outperform the Polar PSO and original PSO algorithm with the rest of modifications; same scenario of performance happened in S3 comparing with all settings in 30 dimensions. In 50 dimensions same performance for all settings and finally in 100 dimensions the performance of modified versions have same results comparing with polar PSO were all outperformed the original PSO. According to all of results in Table 3, the Rosenbrock benchmark test function show that there are no distortion happened in the higher dimensions especially 20, 30, 50 and 100.

Other benchmark test functions: The results mentioned in Table 4-10 were found by applying the same PSO algorithm such described in this study for different benchmark test functions which are Quartic, Dixon, Ackley, DeJong, Griewangk, Rastrigin and Salomon.

Table 8: The performance results for Griewangk test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	5.82e-03	2.45e-02	8.67e-02	2.43e-02	2.27e-02	1.55e-01	7.59e+00
	(5.05e-03)	(1.61e-02)	(4.24e-02)	(2.24e-02)	(4.14e-02)	(6.39e-01)	(1.23e+01)
2	1.61e-01	1.22e-01	1.85e-01	4.64e-01	2.30e+00	2.47e+01	2.35e+02
	(3.73e-01)	(3.63e-01)	(3.47e-01)	(7.88e-01)	(4.60e+00)	(2.44e+01)	(6.20e+01)
3	9.43e-01	1.40e+00	1.54e+00	2.13e+00	4.96e+00	1.00e+02	1.13e+03
	(1.20e+00)	(1.07e+00)	(1.15e+00)	(2.13e+00)	(5.33e+00)	(8.96e+01)	(1.88e+02)

Table 9: The performance results for Rastrigin test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	6.96e-02	5.77e-01	6.04e+00	3.01e+01	6.32e+01	1.66e+02	4.70e+02
	(2.55e-01)	(6.51e-01)	(3.29e+00)	(9.06e+00)	(1.91e+01)	(3.80e+01)	(8.09e+01)
2	4.04e+00	7.18e+00	2.13e+01	6.41e+01	1.16e+02	2.40e+02	6.57e+02
	(5.62e+00)	(5.73e+00)	(8.89e+00)	(1.56e+01)	(2.55e+01)	(3.45e+01)	(8.35e+01)
3	7.06e+00	1.17e+01	2.88e+01	7.87e+01	1.38e+02	2.89e+02	1.42e+03
	(6.00e+00)	(6.58e+00)	(7.82e+00)	(1.53e+01)	(2.18e+01)	(4.87e+01)	(1.85e+02)

Table 10: The performance results for Salomon test function

	PSO setting number		Dimensions				
S	3	5	10	20	30	50	100
1	6.71e-02	9.99e-02	1.49e-01	3.07e-01	5.24e-01	2.16e+00	2.73e+01
	(4.69e-02)	(1.77e-16)	(5.22e-02)	(8.20e-02)	(1.44e-01)	(2.19e+00)	(1.10e+01)
2	6.86e-01	1.78e+00	3.94e+00	7.33e+00	1.19e+01	3.99e+01	1.76e+02
	(1.82e+00)	(3.83e+00)	(3.97e+00)	(3.63e+00)	(3.01e+00)	(2.23e+01)	(3.15e+01)
3	2.89e+00	3.49e+00	5.17e+00	7.84e+00	1.73e+01	8.31e+01	2.32e+02
	(5.22e+00)	(3.81e+00)	(3.97e+00)	(3.04e+00)	(1.13e+01)	(3.11e+01)	(1.51e+01)

According to this, a similar trend observed from the tables except some dimensions setting. Some of the benchmark functions show similar results with Polar PSO and some show better results.

In Table 5, the lower dimension as 3 shows that the Cartesian PSO has better results with constant standard deviation and from dimensions 5-30 the modified versions of polar PSO algorithm show some similarity of the results that obtained in S1, S2. Higher dimensions outperform the Cartesian PSO and similarity with Polar PSO (Table 6-10).

DISCUSSION

This study investigated the polar PSO by reinitialize the particle positions based in polar coordinates initialization where it is distributed more evenly than the normal way of polar initialization in previous work. The significance of use a new way of initialization with Polar PSO which is to make success by removing the distortion that happened. Experimental results showed that the initialization does not effected the PSO and Polar PSO movement to get out of stuck in local optima, previous research mentioned that the problem is caused by the struggling of floating points because of using the transformation from polar to Cartesian coordinates. The movement in Polar PSO is following the Cartesian movement concept weather the particles should move based on the origin point, future direction need to introduce a new model allows the particles to move in polar concepts.

ACKNOWLEDGEMENT

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

CONCLUSION

To overcome the above problem this study introduces a method of particles initialization in polar coordinate for the Polar PSO. Instead of initializing a particle in Cartesian and then transform it to polar the new method starts directly in polar. The initial polar value of a particle is determined by the normal random distribution with some added constrains. The constraints are used to keep the initial values within its search space boundary.

REFERENCES

Bergh, F.V.D., 2002. An analysis of particle swarm optimizers. Ph.D Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa.

Cundy, H. and A. Rollett, 1989. Sphere and Cylinder-Archimedes Theorem. Tarquin Publisher, Stradbroke, England, pp:172-173.

Kendall, M.G., 2004. A Course in the Geometry of n Dimensions. Dover Publications, Mineola, New York, ISBN:0-486-43927-5, Pages: 68.

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proc. IEEE Int. Conf. Neural Networks, 4: 1942-1948.

Kennedy, J. and R.C. Eberhart, 1997. A discrete binary version of the particle swarm algorithm. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Computational Cybernetics and Simulation, Volume 5, October 12-15, 1997, Orlando, FL., USA., pp: 4104-4108.

Matthysen, W. and A.P. Engelbrecht, 2011. A polar coordinate particle swarm optimiser. Applied Soft Comput., 11: 1322-1339.

Pampara, G., A.P. Engelbrecht and N. Franken, 2006. Binary differential evolution. Proceedings of the IEEE Congress on Evolutionary Computation, July 16-21, 2006, IEEE, Vancouver, British Columbia, ISBN: 0-7803-9487-9, pp: 1873-1879.

Pampara, G., N. Franken and A.P. Engelbrecht, 2005. Combining particle swarm optimisation with angle modulation to solve binary problems. Proceedings of the 2005 IEEE Congress on Evolutionary Computation Vol. 1, September 2-5, 2005, IEEE, Edinburgh, Scotland, ISBN: 0-7803-9363-5, pp: 89-96.