ISSN (Online): 2230-7893
www.IJCEM.org

Partitioned PSO: A New Modified Evolutionary Optimization Technique and its Application to Antenna Feed point Calculation

Garg Anju[1], Singh Jagtar[2]
[1, 2] ECE Department, Yadvindra College Of Engg.
Talwandi Sabo, Punjab, India
anjug12@rediffmail.com, jagtarsivian@yahoo.com

Abstract
A new Optimization Technique named as Partitioned Particle Swarm Optimization (PPSO) is proposed. The same is used to determine the feed point position of the planner patch antenna. By dividing the rectangular patch into number of rectangular patch strips, feed point position of the patch antenna modeled as an optimization problem. Computational time to find feed position of patch antenna is reduced considerably. The particles search minutely the sliced search space, thus results in faster convergence and give accurate solution. Sliced Particle swarm optimization tool can be successfully used as soft computing approach in antenna design.

Keywords: *Partitioned Particle Swarm Optimization (PPSO), momentum factor* (mc)*, sliced best (S-best), feed point, cost function.*

1. Introduction

The concept of Particle Swarm Optimization (PSO) was introduced by James Kennedy and Russell C.Eberhart [1] in 1995. PSO is a form of evolutionary computation, stochastic population-based optimization technique. Its search method is inspired by the social and cooperative behavior of various species such as bird flocking, fish schooling and their social behavior. Each potential solution is called particle in PSO system. The particles move with random velocity through the search space to find the optimal solution. The standard PSO has exhibited good performance in solving many optimization problems. PSO tends to suffer from the premature convergence problem; also its performance deteriorates as the dimensionality of search space increases, particularly in case of multimodal optimization problem. Several investigations have been taken by researchers in PSO [2-6] to improve the performance of basic PSO

In basic PSO, particles have memory, and each particle updates its previous best position and corresponding fitness if the current position is better. The previous local best of particle is represented as P_{id} (i^{th} position in *d* dimensions). The global best value of all particles is represented as P_{gd} (i^{th} position in *d* dimensions). In a D-dimensional search space the position of the i^{th} particle is represented by $\overrightarrow{X}_i = (x_{i1}, x_{i2}, ...x_{iD})$. The velocity of each particle is represented as $\overrightarrow{V}_i = (v_{i1}, v_{i2}, ...v_{iD})$. In PSO, a swarm consists of N particles moving around in a D dimensional (whole) search space converges to global best.

The basic particle position and velocity update equations which govern the working of PSO are represented as [1]

$$V_{id} = V_{id} + C_1 \cdot rand() \cdot (P_{id} - X_{id}) + C_2 \cdot rand() \cdot (P_{gd} - X_{id}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

Where C_1 (cognitive component) and C_2 (social component) are two acceleration factors that determine the relative pull for each particle towards p-best and g-best. The $rand()$ is random number between 0 and 1. P_{id} *is* the p-best for i^{th} position in *d* dimensions, which is updated after each iteration. P_{gd} is the g-best at i^{th} position in *d* dimensions, which is updated after each iteration. X_{id} is the i^{th} position of the particles in *d* dimensions and is updated after each iteration as per equation (2). V_{id} is the i^{th} velocity of the particles in *d* dimensions and is updated after each iteration as per equation (1) and is limited in $[V_{\max}, V_{\min}]^D$ .The original PSO has less capability to search the complete space like a full search in a small computational time and no velocity control mechanism.

Eberhart and Shi [7] introduced the time linearly decreasing inertia weight (LDW-PSO) as given in velocity equation (3), which balances between local convergence and global divergence. LDW-PSO gives significant improvement in the performance of basic PSO. The inertia weight descends linearly with increasing iterations.

$$V_{id} = w \cdot V_{id} + C_1 \cdot rand() \cdot (P_{id} - X_{id}) + C_2 \cdot rand() \cdot (P_{gd} - X_{id}) \quad (3)$$

ISSN (Online): 2230-7893
www.IJCEM.org

where

$$w = (w_{final} - w_{initial})(T_{iter} - iter) / T_{iter} + w_{initial} \quad (4)$$

w_{final} and $w_{initial}$ are the higher and lower inertia weight values, respectively. "*iter* " is the current iteration and T_{iter} is the maximum number of iterations. A large inertia weight facilitates a global search while a small inertia weight facilitates a local search.

In Supervisor-Student Model PSO (SSM-PSO) [3] the position update equation is modified by introducing momentum factor (*mc*) and Relaxation-Velocity-Update (RVU) strategy. *mc* restricts the particle in defined search space without checking the boundaries of search space at each iteration. RVU strategy that is to update the velocities of the particles as few times as possible during the run of algorithm is employed to reduce the computational cost for evaluating the velocity. Supervisor-Student model prevents the particles from flying out the search space and RCV strategy balances between global exploration and local exploitation. The position equation of SSM-PSO is expressed as [8]

$$X_{id} = (1 - mc) \cdot X_{id} + mc \cdot V_{id} \quad (5)$$

where mc is the momentum factor (0 < *mc* <1).

In above all discussed methods, the particles move in the search space randomly and may get trapped in local minima. To avoid local trapping the authors propose a modified optimization technique called Partitioned Particle Swarm Optimization (PPSO). In PPSO the search space is sliced and each search space is initialized separately. The number of particles gradually decreases from high to low as the particles move from larger sliced search space to smaller sliced search space. Due to this it reduces the computational cost and gives complete solution.

2. Partitioned Particle Swarm Optimization (proposed by authors)

Partitioned Particle Swarm Optimization (PPSO) algorithms divides the search space into number of rectangular slotted sections that is from outer rectangular slot with higher search space towards inner rectangular slot with less search space. The number of particles also reduces from outer search space towards inner search space. Each search space is initialized separately. Searching the entire search space by slicing gives the complete solution (does not skip even smaller space). It also provides the comparison among all slices, based on Slice best (S-best) of each slice, which gives the complete solution of each search space. Each search space is an independent cluster having own S-best value. The comparison of S-best of each slice leads to g-best or global best.

In PPSO, search is performed more closely in each individual slice. It reduces the chance of getting trapped in local minima and converges to S-best as g-best is decided by comparing S-best not like g-best as in full search. Multiplication of Momentum factor (1 - *mc*) with position gives convergence and multiplication of *mc* with velocity changes its velocity according to position and search the space like a full search. In PPSO the particle position and velocity update equations are expressed as follows

$$V_{id} = w_{varying} \cdot V_{id} + C_1 \cdot rand() \cdot (P_{id} - X_{id}) + C_2 \cdot rand() \cdot (P_{gd} - X_{id}) \quad (6)$$

$$X_{id} = (1 - mc) \cdot X_{id} + mc \cdot V_{id} \quad (7)$$

where

$$w_{varying} = (w_{final} - w_{initial})(T_{iter} - iter) / T_{iter} + w_{initial} \quad (8)$$

mc is the momentum factor $(0 < mc < 1)$,and

$$V_{min} = X_{min} \quad (9)$$

$$V_{max} = X_{max} \quad (10)$$

The $w_{varying}$ has value between 0.4 to 0.9.

The *mc* is taken as 0.45 to bound the particle's minimum and maximum velocity of each sliced search space which is bounded by $[X_{min}, X_{max}]$ of the particular slice. In the proposed method, the velocity and position up gradation is done like SSM-PSO. Particles are defined within the search boundary after each iteration. C_1 and C_2 are taken as 1.

Search space is defined using length (L) and width (W). It is divided into four slices of rectangular shape S_1, S_2, S_3, S_4 . The rectangle slot dimensions P_1, P_2, Q_1, Q_2 are the division of search space along X -direction and R_1, R_2, T_1, T_2 are the division of search space along Y -direction. The slicing can be done using following equations.

$$P_1(I) = L - I \cdot L / 8 \quad (11)$$

$$Q_1(I) = L - (I - 1) \cdot L / 8 \quad (12)$$

$$R_1(I) = W - I \cdot W / 8 \quad (13)$$

$$T_1(I) = W - (I - 1) \cdot W / 8 \quad (14)$$

$$J = 8 - (I - 1) \quad (15)$$

$$P_2(J) = L - J \cdot L / 8 \quad (16)$$

$$Q_2(J) = L - (J - 1) \cdot L / 8 \quad (17)$$

$$R_2(J) = W - J \cdot W / 8 \quad (18)$$

$$T_2(J) = W - (J - 1) \cdot W / 8 \quad (19)$$

ISSN (Online): 2230-7893
www.IJCEM.org

Where the integer $I = 8$, selects the outer rectangle slice S_4 and has $P_1(8)$,$Q_1(8)$ and $P_2(1)$,$Q_2(1)$ lower and upper slot boundaries along X-direction and $R_1(8)$,$T_1(8)$,$R_2(1)$ and $T_2(1)$ lower and upper boundaries along Y-direction. Whenever the particles move beyond the boundaries of the search space (hence it is slice 4), they will be bounded by lower and upper limits of the slot, so that the particle moves in the rectangular slot. Similarly S_1, S_2 and S_3 search space regions can be selected. Also the total number of particles are divided into different slices according to their size of search area. The size of the search area is $S_1 < S_2 < S_3 < S_4$, so S_4 has more particles compared to S_1.The integer I = 5,6,7,8 selects search space i.e. S_1, S_2, S_3, S_4 respectively.

PPSO Algorithm (proposed)

1. Split the search space and divide the particles in each sliced search space
2. Initialize the population of particles in search space of one slice
3. for trials = 1:30

3.1 for iterations = 1: total iterations

3.2 for I = 1:No. of particles

3.3 for J = 1: No. of dimensions

4. Evaluate the fitness of each particle
5. update the pbest position
6. update the gbest position
7. update the position according to formula (7)
8. Bound the position of particle with in boundaries of the slice
9. update the velocity according to formula (6)
10. Bound the velocity of particle with in slice according to equations (9) and (10)
11. find S-best fitness
12. Initialization of each slice is completed

12.2 if No do steps 2-11

12.3 Otherwise, end

13. find g-best fitness

end.

3. Experiment Results on Benchmark functions

3.1 Benchmark Functions

The algorithm is tested on unimodal and multimodal benchmark functions. The two unimodal and multimodal benchmark functions which are used for testing as listed below, minimization criteria is followed.

(1) The Sphere function (unimodal)

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{20}$$

(2) The generalized Schaffer's function (multimodal)

$$f_2(x) = 0.5 - \frac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2} \tag{21}$$

Table 1: Benchmark function

Search Space and Initialization Range		
Function	*Search Space* $[X_{\min}, X_{\max}]$	*Initialization Range*
$f_1(x)$	$-100 \le x_i \le 100$	$50 \le x_i \le 100$
$f_2(x)$	$-100 \le x_i \le 100$	$15 \le x_i \le 30$

The results of two benchmark functions for Partitioned Particle Swarm Optimization technique are good. The mean value of Fitness of the PPSO for each slice is much better. The proposed slicing technique reduces complexity. In basic PSO fitness decreases with increase in dimensions (large search space), reason behind this is fitness is calculated for complete search space, which is the mean contribution of different dimensions. While the value of fitness in a certain dimensions is optimal, however the value in another dimension is very bad. The bad value yield bad overall fitness of the search space. When the search space is sliced, swarm in search space optimize smaller dimension problem and each dimension has large contribution to the final solution. This reduces the chance of losing information even for very small search space. So the overall fitness of lower dimension increases thus reduces the complexity.

Table 2 shows the Mean Fitness Value of 30 trials for the Sphere function (unimodal) for S-best of each slice of PPSO and minimum value of fitness of each slice. Similarly the Table 3 shows the Schaffer's function (two dimensional). The selected mean S_1, S_2, S_3, S_4 represents the S-best value of each slice and the selected mean 'S' represents the g-best value of each slice i.e. S-best. For the Sphere function, fig. 1 to 3 depicts the S-best fitness of different slices and g-best fitness verses' number of trials for varied particle size. Fig. 4 shows the convergence for 40 particles with 10 dimensions converges fast, gives g-best fitness. For the Schaffer's function, fig. 5 to 7 depicts the S-best fitness of different slices and g-best fitness verses' number of trials for varied particle size. Fig. 8 shows convergence for 40 particles with 20 dimensions converges fast, gives g-best fitness.

ISSN (Online): 2230-7893
www.IJCEM.org

Table 2: Sphere function

Mean Fitness Value for the Sphere function							
Population size	***Dimensions***	***Max Iterations***	***Mean*** S_4	***Mean*** S_3	***Mean*** S_2	***Mean*** S_1	***Mean*** S
20	10	1000	80.6875	5.0066	25.8914	6.6213	0.2334
20	20	1500	112.9138	2.2286	7.7737	34.0624	0.0904
20	30	2000	247.4600	1.8318	58.7163	20.3452	0.0782
40	10	1000	30.0598	9.4939	26.1020	0.6019	7.0428e-004
40	20	1500	74.8393	9.6026	2.6420	5.6981	0.2418
40	30	2000	133.9569	0.0821	1.4817	8.6145	0.0821
80	10	1000	11.4541	4.3009	20.5775	2.2406	0.1251
80	20	1500	36.0802	11.5666	62.2024	4.8604	0.1747
80	30	2000	56.5048	0.0783	31.4297	13.2500	0.0723

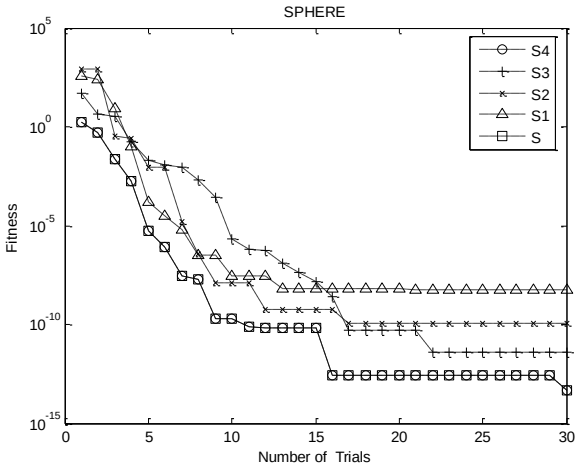

Fig. 1 Results for 20 Particles and 30 Dimensions.

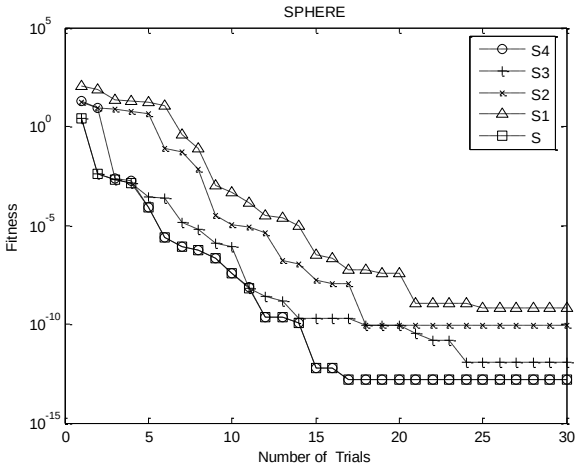

Fig. 2 Results for 40 Particles and 30 Dimensions.

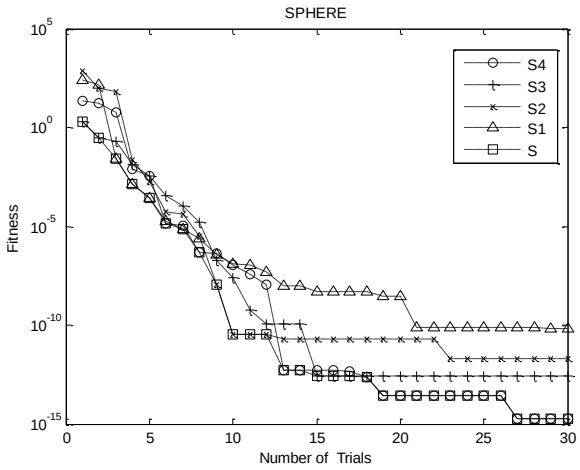

Fig. 3 Results for 80 Particles and 30 Dimensions.

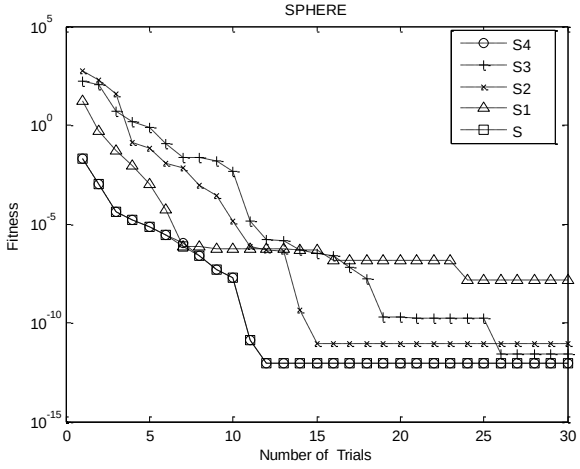

Fig. 4 Results for 40 Particles and 10 Dimensions.

ISSN (Online): 2230-7893
www.IJCEM.org

Table 3: Schaffer's function .

Mean Fitness Value for the Schaffer's function							
Population size	***Dimensions***	***Max Iterations***	***Mean*** S_4	***Mean*** S_3	***Mean*** S_2	***Mean*** S_1	***Mean*** S
20	10	1000	0.0599	0.0250	0.0694	0.0591	0.0198
20	20	1500	0.0262	0.0339	0.0766	0.1261	0.0218
20	30	2000	0.1408	0.0777	0.0547	0.0465	0.0416
40	10	1000	0.0879	0.0452	0.0483	0.0735	0.0342
40	20	1500	0.0317	0.0675	0.0302	0.0206	0.0133
40	30	2000	0.0465	0.0576	0.0775	0.0744	0.0323
80	10	1000	0.0355	0.0181	0.0788	0.0372	0.0174
80	20	1500	0.0469	0.0508	0.0588	0.0334	0.0251
80	30	2000	0.0609	0.0431	0.0691	0.0955	0.0284

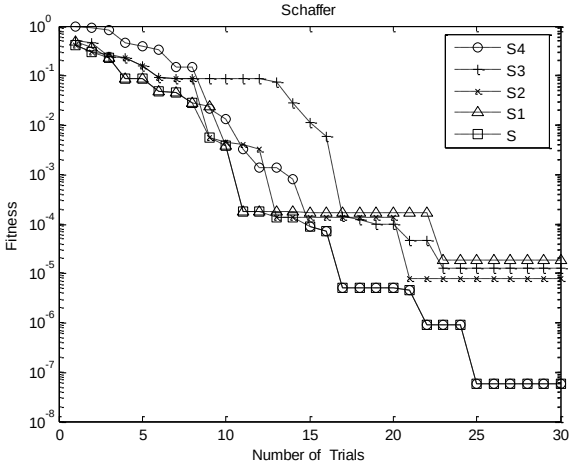

Fig. 5 Results for 20 Particles and 30 Dimensions.

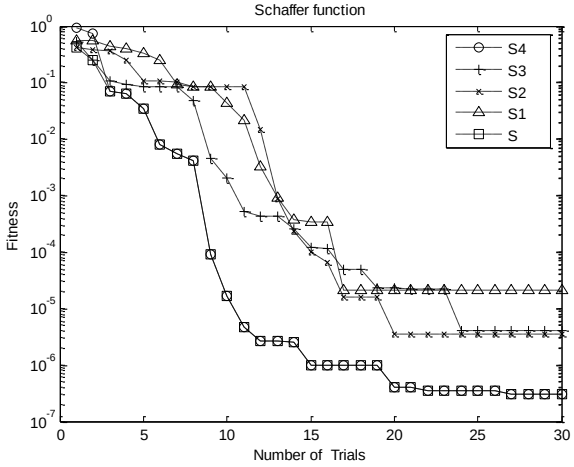

Fig. 7 Results for 80 Particles and 30 Dimensions.

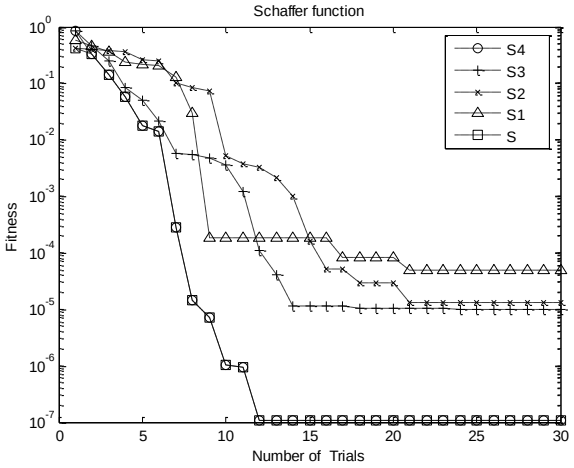

Fig. 6 Results for 40 Particles and 30 Dimensions

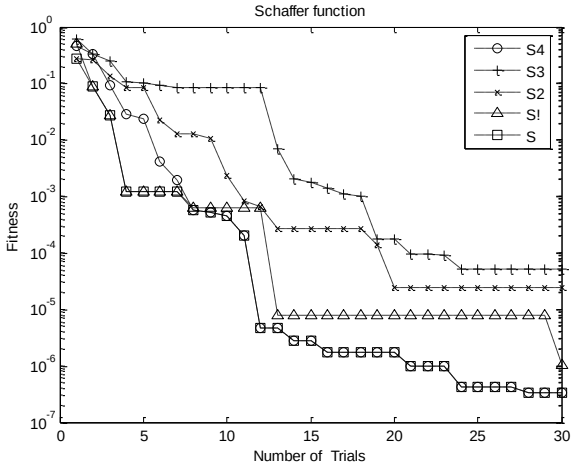

Fig. 8 Results for 40 Particles and 20 Dimensions.

ISSN (Online): 2230-7893
www.IJCEM.org

Table 4: planner patch antenna

Feed Position of the planner patch antenna								
	Population size	***Max Iterations***	***Initial position along X-axis***	***Initial position along Y-axis***	$Z_{inactual}$	***X position***	***Y position***	Z_{error}
S_1	16	110000	rand * 79.275/2	103.77/2	18.015	37.089	48.552	-31.985
S_2	16	110000	rand * 56.625/2	74.125/2	50.0000	31.2569	33.727	-2.0354e-005
S_3	16	110000	rand * 33.975/2	44.475/2	49.9997	31.2569	18.902	-0.00030682
S_4	16	110000	rand * 11.325/2	14.825/2	175.3024	14.5037	4.0769	125.3
COMP SLICE	16	290000	rand*90.6*2/8	118.6/2	50.0000	31.257	32.618	-1.038e-005

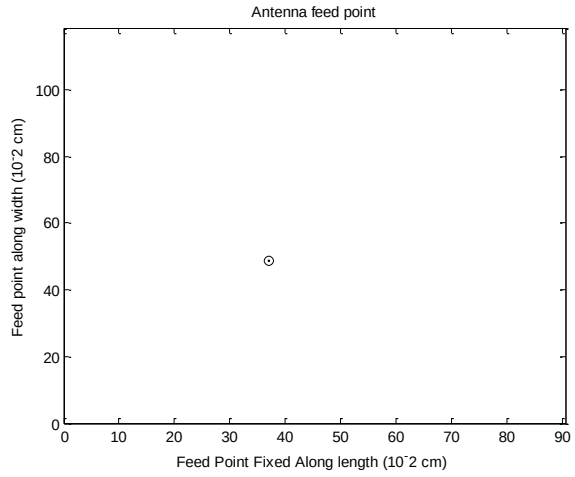

Fig . 09 Feed point position for S1 Slice.

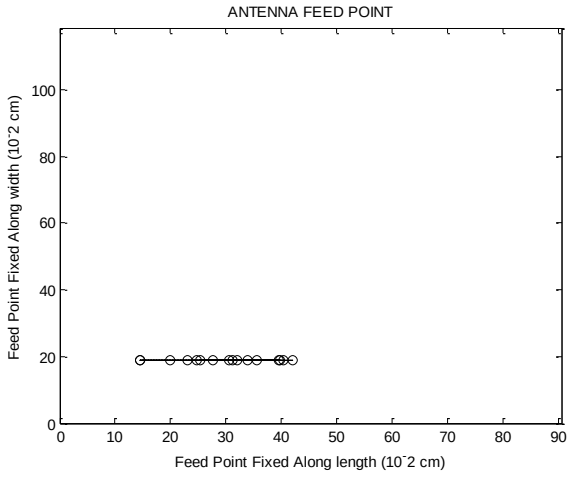

Fig.11 Feed point position for S3 Slice.

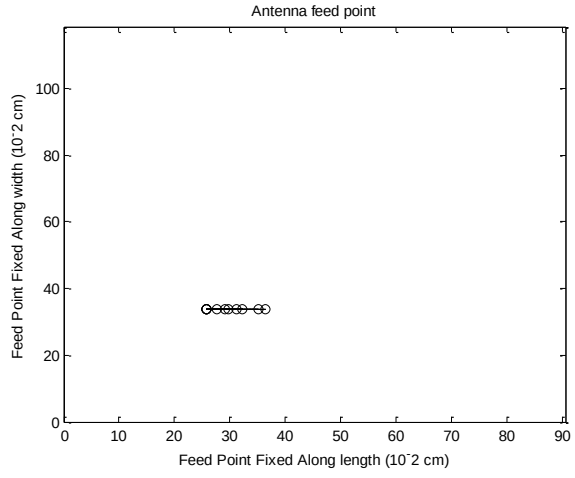

Fig. 10 Feed point position for S2 Slice .

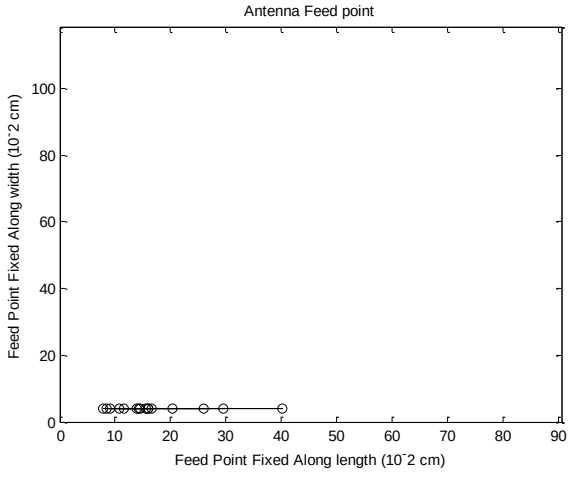

Fig.12 Feed point position for S4 Slice.

ISSN (Online): 2230-7893
www.IJCEM.org

3.2 Feed Point Calculation of Planner Patch Antenna Using PPSO

The Cost function used to calculate the feed point position of the planner patch is expressed as[ ]

$$Z_{inactual} = Z_i \cdot (\cos^2(\pi \cdot \frac{Y_0}{Length})) \quad (22)$$

Where $Z_i = Input\ \mathrm{Im}\ pedence$

Y_0 is the feed point position to patch antenna

Length = 0.906 cm
Width = 1.186 cm
Height = 0.1588 cm
Dielectric constant = 2.2 and
Operating freq. at 10 GHZ
Input Impedance = 228.3508 ohm
Desired Input Impedance = 50 ohm

Feed point of patch antenna for the full patch which takes higher number of iteration in comparison with the sliced patch antenna. Accuracy in feed point calculation is very high in sliced patch antenna in comparison with full patch antenna search. Table 4 show the Planner patch Antenna for different slice of patch antenna.
For the Planner patch antenna, fig. 9 to 12 depicts the S-best fitness of different slices and feed point along X-axis verses' feed point along Y-axis. The stopping criterion of the algorithm is fixed at error below 0.001. Fig. 9 shows the feed point of patch antenna for the inner most rectangular slice S_1, fig. 11 shows the feed point of patch antenna for the second outermost slice S_3 and fig. 12 shows the feed point of patch antenna for the outer most slice S_4, in which the simulation of algorithm reaching to maximum number of iterations and does not find the optimized value of input impedance. But fig. 10 shows the feed point of patch antenna for the Third outer most rectangular slice S_2, in which the simulation of algorithm reaches error below 0.001 and find the optimized value of input impedance. Fig. 13 shows the feed point of patch antenna for the full patch which takes higher number of iteration in comparison with the sliced patch antenna search technique.

4. Conclusions

The paper presents a Partitioned Particle swarm optimization (PPSO) algorithm. The proposed PPSO is computationally efficient as it follows a sliced search strategy that gives a complete solution of full search without over loading the computational time. Thus the accuracy also increases. The testing on benchmark functions shows strength of the proposed new algorithm. The application to feed point application opens up a pathway to use evolutionary computation for various antenna parameter calculations. The proposed algorithm will find potential applications in antenna, wireless communication, body centre devices and in medical image analysis and diagnosis.

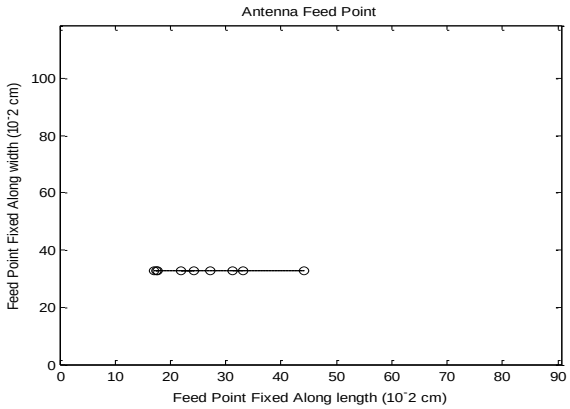

Fig. 13 Feed point position of Full patch

References

[1] J. Kenedy, R. C.Eberhart," particle swarm optimization, "Proceedings of IEEE International Conference on Neural Networks,Piscataway,NJ.pp.1942-1948,1995.
[2] Jing Jie,Jianchao Zeng," Particle Swarm Optimization with Diversity-controlled Acceleration Coefficients," IEEE Third International Conference on Neural Computation(ICNC),0-7695-2875-9/07,2007.
[3] Jang-Ho Seo,Chang-Hwan Im,Chang-Geun Heo," Multimodal Function Optimization based on Particle Swarm Optimization," IEEE Transactions of magnetics,Vol.42,NO,4,April,2006.
[4] Nanbo Jin, Yahya Rahmat -Damii, "Parallel PSO and Finite Difference Time Domain Algorithm for Multiband and Wide Band Patch antenna Design, "IEEE Transaction on antenna and propagation, vol. 53,no.11,pp 3468-3468,2005.
[5] H. Liu, A Abraham and W. Zhang, "A Fuzzy Adaptive Turbulent particle swarm optimization, " International Journal of Innovative Computing and Applications, Volume 1, Issue 1, pp. 39-47, 2007.
[6] Pant, R. Thangaraj and A. Abraham, "PSO: Performance Tuning and Empirical Analysis, "Foundations of Computational Intelligence Volume 3: Global Optimization, Studies in Computational Intelligence, Springer Verlag, Germany, ISBN: 978-3-642-01084-2, pp. 101-128, 2009.
[7] Y Shi, R. C. Eberhart, "A modified particle swarm optimizer, "proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway,NJ.pp.69-73,1998.
[8] Yu. Liu, Zheng Quin, Xingshi He, "Supervisor-Student Model in PSO," proceedings of 2004 IEEE congress on Evolutionary Computation, pp.542-547, 2004.