Improved Particle Swarm Optimization with Low-Discrepancy Sequences

Millie Pant, Radha Thangaraj, Crina Grosan, and Ajith Abraham

Abstract— **Quasirandom or low discrepancy sequences, such as the Van der Corput, Sobol, Faure, Halton (named after their inventors) etc. are less random than a pseudorandom number sequences, but are more useful for computational methods which depend on the generation of random numbers. Some of these tasks involve approximation of integrals in higher dimensions, simulation and global optimization. Sobol, Faure and Halton sequences have already been used [7, 8, 9, 10] for initializing the swarm in a PSO. This paper investigates the effect of initiating the swarm with another classical low discrepancy sequence called Vander Corput sequence for solving global optimization problems in large dimension search spaces. The proposed algorithm called VC-PSO and another PSO using Sobol sequence (SO-PSO) are tested on standard benchmark problems and the results are compared with the Basic Particle Swarm Optimization (BPSO) which follows the uniform distribution for initializing the swarm. The simulation results show that a significant improvement can be made in the performance of BPSO, by simply changing the distribution of random numbers to quasi random sequence as the proposed VC-PSO and SO-PSO algorithms outperform the BPSO algorithm by noticeable percentage, particularly for problems with large search space dimensions.**

I. Introduction

Particle swarm optimization technique is one of the most promising tools for solving global optimization problems. It is a population based stochastic search technique first suggested by Kennedy and Eberhart in 1995 [1]. Because of its simplicity and robustness, it immediately became a popular technique for solving complex optimization problems arising in various diversified fields of science and engineering.

PSO (and other search techniques which depend on the generation of random numbers) works very well for problems having a small search area (i.e. a search area having low dimension), but as we go on increasing the dimension of search space the performance deteriorates and many times converge prematurely giving a suboptimal result [2]. This problem becomes more persistent in case of multimodal functions having several local and global optima. One of the reasons for the poor performance of a PSO may be attributed to the dispersion of initial population points in the search space i.e. to say, if the swarm population does not cover the search area efficiently, it may not be able to locate the potent solution points, thereby missing the global optimum [3]. This difficulty may be minimized to a great extent by selecting a well-organized distribution of random numbers.

The most common practice of generating random numbers is the one using an inbuilt subroutine (available in most of the programming languages), which uses a uniform probability distribution to generate random numbers. This method is not very proficient as it has been shown that uniform pseudorandom number sequences have discrepancy of order $(\log(\log N))^{1/2}$ and thus do not achieve the lowest possible discrepancy. Subsequently, researchers have proposed an alternative way of generating 'quasirandom' numbers through the use of low discrepancy sequences. Their discrepancies have been shown to be optimal, of order $(\log N)^s/N$ [4], [5]. Quasirandom sequences, on the other hand are more useful for global optimization, because of the variation of random numbers that are produced in each iteration.

In case of population based search algorithms like Evolutionary Algorithms, Genetic algorithms, Particle Swarm Optimization etc., not much research has been done on the use of quasi random sequences. Some previous instances where low discrepancy sequences have been used to improve the performance of optimization algorithms include [6, 7, 8, 9, 10]. Kimura and Matsumura [6] have used Halton sequence for initializing the Genetic Algorithms (GA) population and have shown that a real coded GA performs much better when initialized with a quasi random sequence in comparison to a GA which initialized with a population having uniform probability distribution. Instances where quasi random sequences have been used for initializing the swarm in PSO can be found in [7, 8, 9, 10]. In [8, 9, 10] authors have made use of Sobol and Faure sequences. Similarly, Nguyen et al. [7] have shown a detailed comparison of Halton Faure and Sobol sequences for initializing the swarm. In the previous studies, it has already been shown that the performance of Sobol sequence dominates the performance of Halton and Faure sequences. However to the best of our knowledge, no results are available on the performance of Van der Corput sequence which is a well known sequence and forms the basis of many other sequences.

M. Pant is with the Indian Institute of Technology Roorkee, Saharanpur, 247001, India (phone: +91-9759561464; e-mail: millifpt@iitr.ernet.in).

T. Radha is with the Indian Institute of Technology Roorkee, Saharanpur, India (e-mail: radhadpt@iitr.ernet.in).

C. Grosan is with the Babes – Bolyai University, Cluj-Napaco, Romania. (e-mail: cgrosan@cs.ubbcluj.ro)

A. Abraham is with the Center of Excellence for Quantifiable Quality of Service, Norwegian University of Science and Technology, Norway (e-mail: ajith.abraham@ieee.org).

Many of the relevant low discrepancy sequences are linked to the Van der Corput sequence introduced initially for dimension s = 1 and base b = 2 [11]. The Van der Corput discovery inspired other quasi random sequences like Halton [12], Faure, Sobol [13] [14], etc. However, it has been reported that Halton and Faure sequences do not work too well when the search space has large dimensions. Keeping this fact in mind we decided to scrutinize the performance of PSO using Van der Corput sequence along with Sobol sequence (which is said be superior than other low discrepancy sequences according to the previous studies) for swarm initialization and tested them for solving global optimization problems in large dimension search spaces.

The remaining paper is organized as follows: in Section II, we have briefly described the BPSO algorithm. Section III describes the Vander Corput and Sobol sequences along with the proposed VC-PSO and SO-PSO algorithm. The experimental setup, parameter settings and benchmark problems are reported in Section IV. The experimental results are analyzed in Section V, finally the paper concludes with Section VI.

II. Basic Particle Swarm Optimization

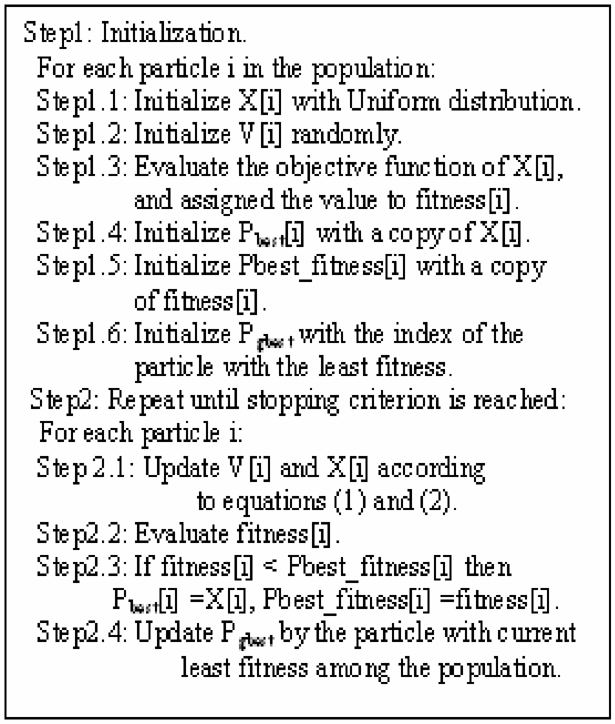
Particle Swarm Optimization (PSO) is a relatively newer addition to a class of population based search technique for solving numerical optimization problems. Its mechanism is inspired from the complex social behavior shown by the natural species like flock of birds, school of fish and even crowd of human beings. The particles or members of the swarm fly through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time as per its own experience and also as per the position of its neighbors (or colleagues).

For a D-dimensional search space the position of the ith particle is represented as $X_i = (x_{i1}, x_{i2},.., x_{iD})$. Each particle maintains a memory of its previous best position $P_i = (p_{i1}, p_{i2}\ldots p_{iD})$ and a velocity $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ along each dimension. At each iteration, the P vector of the particle with best fitness in the local neighborhood, designated g, and the P vector of the current particle are combined to adjust the velocity along each dimension and a new position of the particle is determined using that velocity. The two basic equations which govern the working of PSO are that of velocity vector and position vector are given by:

$$v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

The first part of equation (1) represents the inertia of the previous velocity, the second part is the cognition part and it tells us about the personal thinking of the particle, the third part represents the cooperation among particles and is therefore named as the social component [15]. Acceleration constants c_1, c_2 [16] and inertia weight ω [17] are the predefined by the user and r_1, r_2 are the uniformly generated random numbers in the range of [0, 1]. The computational steps of BPSO algorithm are shown in Figure 1.



```
Step1: Initialization.
  For each particle i in the population:
  Step1.1: Initialize X[i] with Uniform distribution.
  Step1.2: Initialize V[i] randomly.
  Step1.3: Evaluate the objective function of X[i],
           and assigned the value to fitness[i].
  Step1.4: Initialize $P_{best}$[i] with a copy of X[i].
  Step1.5: Initialize Pbest_fitness[i] with a copy
           of fitness[i].
  Step1.6: Initialize $P_{gbest}$ with the index of the
           particle with the least fitness.
Step2: Repeat until stopping criterion is reached:
  For each particle i:
  Step2.1: Update V[i] and X[i] according
           to equations (1) and (2).
  Step2.2: Evaluate fitness[i].
  Step2.3: If fitness[i] < Pbest_fitness[i] then
           $P_{best}$[i] =X[i], Pbest_fitness[i] =fitness[i].
  Step2.4: Update $P_{gbest}$ by the particle with current
           least fitness among the population.
```

Fig. 1 Flow of BPSO algorithm

III. Proposed Algorithms

As stated above, the purpose of this paper is to test the integrity of quasi random (or low discrepancy) sequences (for the present study the chosen sequences are Van der Corput and Sobol) for generating the initial population of swarm in a PSO algorithm. Mathematically, the discrepancy of a sequence is the measure of its uniformity which may be defined as follows:

For a given set of points $x^1, x^2, \ldots, x^N \in I^S$ and a subset $G \subset I^S$, define a counting function

$S_N(G)$ as the number of points $x^i \in G$

For each $x = (x_1, x_2, \ldots. x_S) \in I^S$,

Let G_x be the rectangular S dimensional region.

$G_x = [0,x_1) \times [0,x_2) \times \ldots \times [0,x_S)$

With volume $x_1x_2\ldots x_N$. Then the discrepancy of points is given by $D^*_N(x^1, x^2, x^3 \ldots. x^N) = \text{Sup}\ |S_N(G_x) - Nx_1x_2\ldots x_S|$, $x \in I^S$.

The discrepancy is therefore computed by comparing the actual number of sample points in a given volume of a multi-dimensional space with the number of sample points that should be there assuming a uniform distribution. Figures 2 and 3 show a pseudo random sequence and a quasi random sequence. From these figures it can be seen easily that the distribution of points is more systematic in a quasi random sequence in comparison to a pseudo random sequence.

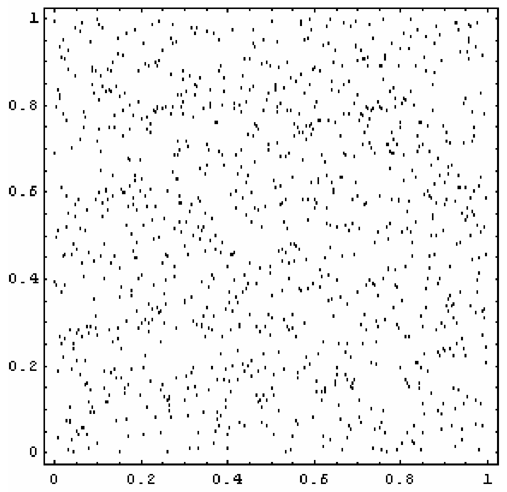


Fig. 2 Sample points generated using a pseudo random sequence

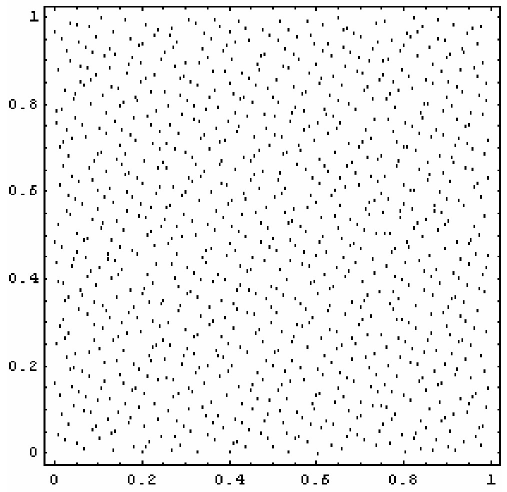


Fig. 3 Sample points generated using a quasi random sequence

A. Van der Corput Sequence

A Van der Corput sequence is a low-discrepancy sequence over the unit interval first published in 1935 by the Dutch mathematician J. G. Van der Corput. It is a digital (0, 1)-sequence, which exists for all bases $b \geq 2$. It is defined by the *radical inverse function* $\varphi_b : N_0 \rightarrow [0, 1)$. If $n \in N_0$ has the b-adic expansion

$$n = \sum_{j=0}^{T} a_j b^{j-1} \tag{3}$$

with $a_j \in \{0,\dots, b-1\}$, and $T = \lfloor \log_b n \rfloor$ then φ_b is defined as

$$\varphi_b(n) = \sum_{j=0}^{T} \frac{a_j}{b^j} \tag{4}$$

In other words, the jth b-adic digit of n becomes the jth b-adic digit of $\varphi_b(n)$ behind the decimal point. The Van der Corput sequence in base b is then defined as $(\varphi_b(n))_{n \geq 0}$.

The elements of the Van der Corput sequence (in any base) form a dense set in the unit interval: for any real number in [0, 1] there exists a sub sequence of the Van der Corput sequence that converges towards that number. They are also uniformly distributed over the unit interval.

B. Sobol Sequence

The construction of the Sobol sequence [18] uses linear recurrence relations over the finite field, F2, where F2 = {0, 1}. Let the binary expansion of the non-negative integer n be given by $n = n_1 2^0 + n_2 2^1 + \dots + n_w 2^{w-1}$. Then the nth element of the jth dimension of the Sobol Sequence, $X_n^{(j)}$, can be generated by:

$$X_n^{(j)} = n_1 v_1^{(j)} \oplus n_2 v_2^{(j)} \oplus \dots \oplus n_w v_w^{(j)}$$

where $v_i^{(j)}$ is a binary fraction called the ith direction number in the jth dimension. These direction numbers are generated by the following q-term recurrence relation:

$$v_i^{(j)} = a_1 v_{i-1}^{(j)} \oplus a_2 v_{i-2}^{(j)} \oplus \dots \oplus a_q v_{i-q+1}^{(j)} \oplus v_{i-q}^{(j)} \oplus (v_{i-q}^{(j)} / 2^q)$$

We have i > q, and the bit, a_i, comes from the coefficients of a degree-q primitive polynomial over F2.

C. VC-PSO and SO-PSO Algorithm

It has been shown that uniformly distributed particles may not always be good for empirical studies of different algorithms. The uniform distribution sometimes gives a wrong impression of the relative performance of algorithms as shown by Gehlhaar and Fogel [19].

The quasi random sequences on the other hand generates a different set of random numbers in each iteration, thus providing a better diversified population of solutions and thereby increasing the probability of getting a better solution.

Keeping this fact in mind we decided to use the Vander Corput sequence and Sobol sequence for generating the swarm. The swarm population follows equation (1) and (2) for updating the velocity and position of the swarm. However for the generation of the initial swarm Van der Corput Sequence and Sobol Sequences have been used for VC-PSO and SO-PSO respectively.

IV. Experimental Settings

For all the algorithms, a linearly decreasing inertia weight is used which starts at 0.9 and ends at 0.4, with the user defined parameters $c_1 = c_2 = 2.0$ and r_1, r_2 as uniformly distributed random numbers between 0 and 1. For each function, four different dimension sizes of 10, 20, 30 and 50 are taken. The maximum number of generations is set as 1000, 1500 and 2000 with population sizes of 20, 40 corresponding to the dimensions 10, 20, 30 and 50 respectively. Stopping criteria for all the algorithms is taken as the maximum number of generations. A total of 30 runs for each experimental setting are conducted and the average fitness of the best solutions throughout the run is recorded. We have taken two different ranges R1 and R2 for the search space for all the test functions.

V. Test Functions and Experimental Results

In order to check the compatibility of the proposed VC-PSO and SO-PSO algorithms we took a test suite of four unconstrained, classical bench mark functions, given in Table I, that are often used for deciding the credibility of an optimization algorithm. Functions f_1, f_2, and f_4 are highly multimodal in nature. Moreover, we have taken two different ranges R1 and R2 for the search space for all the test functions. R_1 is the original dimension of the search space that is generally cited in the research papers dealing with the comparison of optimization algorithms, but since in this paper we are checking the credibility of quasi random sequences to produce good solution points in a large dimension search space, we took another range R2 with increased dimension of the original search space to allow for comparison on more difficult problem instances. The results of BPSO algorithm corresponding to the range R1 for function f1, f2 and f3 are taken from [20]. The mean best fitness value for the functions f_1 – f_5 are given in Tables II – VI, respectively, in which P represents the swarm population, D represents the dimension and G represents the maximum number of permissible generations. Figures 4, 5, 6 and 7 show the mean best fitness curves for the given functions.

TABLE I
NUMERICAL BENCHMARK PROBLEMS

Function	Optimum	Range 1 (R1)	Range 2 (R2)
$f_1(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	0	[-5.12,5.12]	[-100,100]
$f_2(x)=\frac{1}{4000}\sum_{i=0}^{n-1}x_i^2-\sum_{i=0}^{n-1}\cos(\frac{x_i}{\sqrt{i+1}})+1$	0	[-600,600]	[-1000,1000]
$f_3(x)=\sum_{i=0}^{n-1}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	0	[-30,30]	[-100,100]
$f_4(x)=20+e-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$	0	[-32,32]	[-100,100]
$f_5(x)=\max\lvert x_i\rvert,\quad 0\le i<n$	0	[-100,100]	[-500,500]

TABLE II
RESULT OF TEST FUNCTION f_1

P	D	G	R1			R2		
			BPSO	SO-PSO	VC-PSO	BPSO	SO-PSO	VC-PSO
20	10	1000	5.5572	4.647559	**3.880492**	6.785225	5.919983	**2.985324**
	20	1500	22.8892	22.213901	**19.615505**	34.547704	15.7203	**12.743894**
	30	2000	47.2941	41.613066	**40.793075**	66.816699	26.023739	**25.172492**
	50	3000	105.465	88.211285	**81.509773**	187.74240	53.38977	**42.797887**
40	10	1000	3.5623	**2.984863**	3.351035	6.133424	2.935168	**2.737171**
	20	1500	16.3504	16.021596	**15.919263**	24.27542	13.026489	**12.541202**
	30	2000	38.5250	34.684111	**31.755918**	51.027076	25.474607	**19.322985**
	50	3000	92.8400	66.018642	**62.149**	137.38657	45.026485	**40.150249**

TABLE III

RESULT OF TEST FUNCTION f_2

P	D	G	R1			R2		
			BPSO	SO-PSO	VC-PSO	BPSO	SO-PSO	VC-PSO
20	10	1000	0.0919	**0.001229**	0.003326	0.149345	0.056995	**0.004313**
	20	1500	0.0313	**0.000986**	0.002583	0.041509	7.819e-16	**8.956e-15**
	30	2000	0.0182	**7.651e-12**	0.000986	0.01633	2.777e-11	4.787e-11
	50	3000	0.0147	2.162e-09	**1.849e-09**	0.018478	4.849e-09	**3.074e-07**
40	10	1000	0.0862	**0.001725**	0.005169	0.090361	0.008491	**0.003446**
	20	1500	0.0286	**8.674e-20**	0.003194	0.037427	**1.626e-19**	2.240e-16
	30	2000	0.0127	3.284e-14	**3.064e-14**	0.018426	**4.941e-14**	8.669e-14
	50	3000	0.0098	5.837e-11	**3.772e-11**	0.008983	**1.353e-10**	5.837e-11

TABLE IV

RESULT OF TEST FUNCTION f_3

P	D	G	R1			R2		
			BPSO	SO-PSO	VC-PSO	BPSO	SO-PSO	VC-PSO
20	10	1000	96.1715	4.233931	**3.650252**	27.931384	**4.165369**	4.243705
	20	1500	214.6764	16.288114	**13.44568**	102.43696	17.218995	**11.485271**
	30	2000	316.4468	33.615175	**32.399337**	159.80824	**39.181265**	45.264272
	50	3000	533.648	86.268354	**84.536223**	210.43212	**85.600516**	89.927502
40	10	1000	70.2139	**3.908925**	4.041561	27.845406	**3.799519**	4.064811
	20	1500	180.9671	15.856598	**12.403083**	56.356471	**12.625576**	16.373076
	30	2000	299.7061	36.66065	**31.897961**	148.85530	38.830241	**33.646456**
	50	3000	482.135	83.583921	**81.065624**	232.98606	88.173599	**73.407326**

TABLE V

RESULT OF TEST FUNCTION f_4

P	D	G	R1			R2		
			BPSO	SO-PSO	VC-PSO	BPSO	SO-PSO	VC-PSO
20	10	1000	6.965e-12	**2.996e-12**	3.114e-12	20.025405	**3.081e-11**	9.764e-12
	20	1500	3.560e-07	**6.681e-08**	8.572e-08	19.999566	1.946e-07	**1.619e-07**
	30	2000	3.618e-05	8.692e-06	**2.649e-06**	20.010351	**1.399e-05**	2.408e-05
	50	3000	3.43866	5.690e-04	**2.024e-03**	20.025072	**1.433e-03**	1.497e-03
40	10	1000	7.897e-13	**3.495e-14**	5.608e-14	16.334716	**2.558e-13**	7.782e-14
	20	1500	5.045e-08	**2.509e-09**	2.654e-09	19.999332	9.823e-09	**3.452e-09**
	30	2000	7.269e-06	6.693e-07	**4.418e-07**	20.049032	1.166e-06	**3.975e-07**
	50	3000	1.966554	**1.090e-05**	1.371e-04	20.030038	9.158e-05	**7.749e-05**

TABLE VI

RESULT OF TEST FUNCTION f_5

P	D	G	R1			R2		
			BPSO	SO-PSO	VC-PSO	BPSO	SO-PSO	VC-PSO
20	10	1000	4.831e-05	4.420e-06	**2.210e-11**	5.736e-05	5.046e-06	**4.420e-06**
	20	1500	1.501872	0.086019	**0.069028**	1.564932	0.099365	**0.086019**
	30	2000	10.918735	**0.437305**	0.443966	9.519843	**0.420949**	0.437305
	50	3000	33.56714	**0.802782**	0.828851	33.430501	0.811536	**0.802782**
40	10	1000	7.421e-07	5.894e-08	**4.651e-15**	6.235e-07	8.414e-08	**5.893e-08**
	20	1500	0.343671	0.020741	**0.016538**	0.368743	0.026471	**0.020741**
	30	2000	7.481037	**0.280413**	0.32125	7.102399	0.331953	**0.280413**
	50	3000	29.9286	**0.508336**	0.706595	26.658801	**0.786058**	0.808336

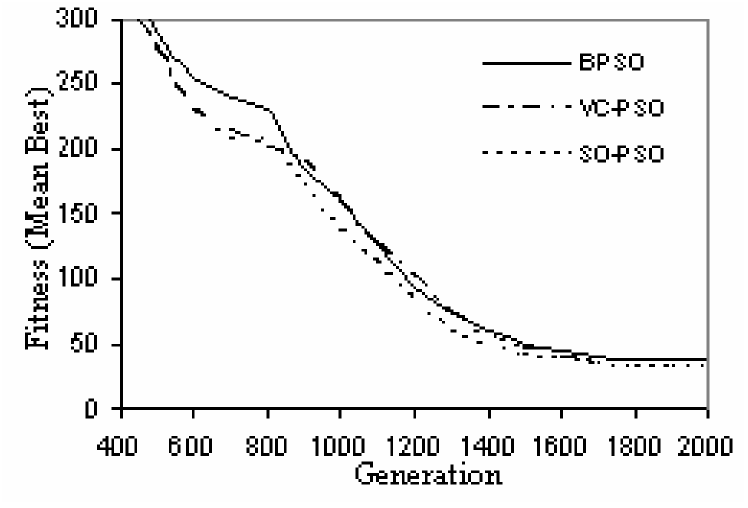


Fig. 4. Convergence graph for function f_1

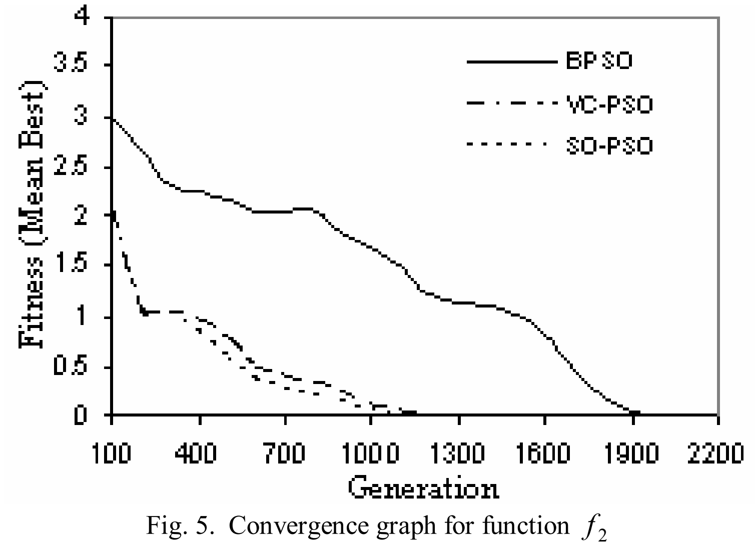


Fig. 5. Convergence graph for function f_2

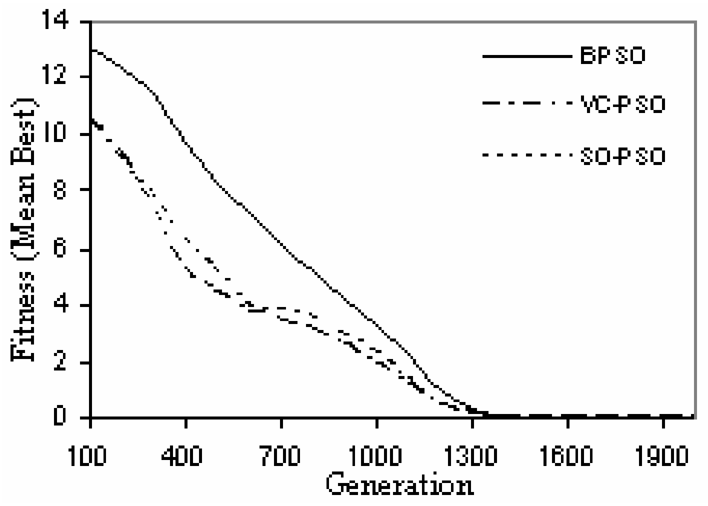


Fig. 6. Convergence graph for function f_4

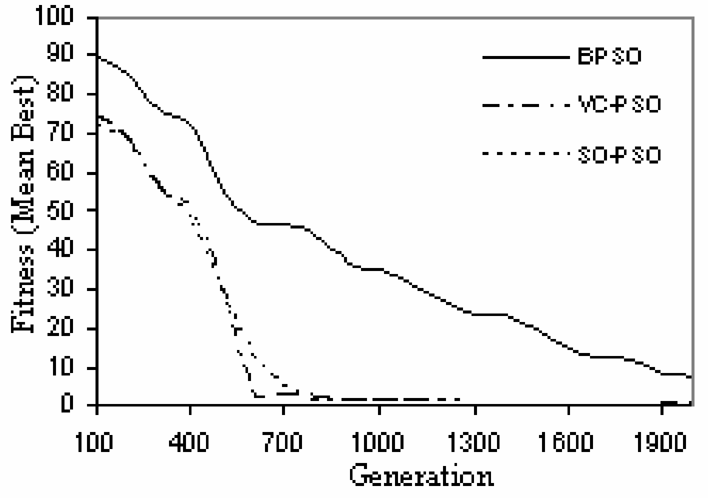


Fig. 7. Convergence graph for function f_5

VI. Discussions and Conclusion

The present study inspects the performance of two common Low Discrepancy Sequences namely Van der Corput sequence and Sobol Sequences for swarm initialization in PSO, for solving global optimization problems. Their performance is compared with the BPSO algorithm on problems with varying dimensions of 10, 20, 30 and 50. Our particular interest was to see the performance of these algorithms for problems having a large search space. Therefore for all the test problems we have taken different dimensions of the search space R1 and R2 ranging from [-5.12, 5.12] to [-1000, 1000].

The numerical results show that although VC-PSO and SO-PSO gives a better performance than BPSO for all the test problems, their performance is much better for problems with large search space (functions f2 to f5). In the first function where the dimensions of search space is small [-5.12, 5.12], the percentage of improvement in the mean fitness value is not quite evident. However as we increase the dimension of the search space to [-100,100], the superior performance of VC-PSO and SO-PSO becomes more obvious. In the second function, when the dimension of the search space is [-600,600], the performance of all the algorithms is satisfactory in comparison o the true optimum (which is 0). But as we increase the dimension to [-1000, 1000], VC-PSO and SO-PSO performs much better than BPSO. Similarly for f3, f4 and f5, we can easily judge from the numerical results the significant superiority of VC-PSO and SO-PSO over BPSO for both the ranges.

If we compare VC-PSO and SO-PSO with each other we see that out of the 80 cases (40 cases for R1 and 40 cases for R2) tested in this study VC-PSO gave a better performance in 24 cases (60 %) and SO-PSO gave a better performance in 16 cases (40%). For the larger search space (R2), VC-PSO gave a better mean function value in 25 cases (62.5%) and SO-PSO in remaining 15 cases (37.5%).

From this study we can conclude that quasi random sequences like Vander Corput and Sobol are much better for generating random numbers for PSO and most probably for all the population search algorithms. In future, we plan to work for problems with larger dimensions and also constrained optimization problems. Also in this study we have not used any additional operators like mutation etc. and it will be interesting to see the effect of these operators on VC-PSO and SO-PSO.

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", IEEE Int. Conf. on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, 1995, pp. 1942-1948.

[2] 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, 2007, pp. 39-47.

[3] C. Grosan, A. Abraham and M. Nicoara, "Search Optimization Using Hybrid Particle Sub-Swarms and Evolutionary Algorithms", International Journal of Simulation Systems, Science & Technology, UK, Volume 6, Nos. 10 and 11, 2005, pp. 60-79.

[4] E. J. Gentle, "RandomNumber Generation and Monte Carlo Methods", Springer-Verlog, 1998.

[5] D. E. Knuth, "The Art of Computer Programming", Semi numerical Algorithms, Vol. 2, Addison-Wesley, 1998.

[6] S. Kimura and K. Matsumura, "Genetic Algorithms using low discrepancy sequences", in proc of GEECO 2005, pp. 1341 – 1346.

[7] Nguyen X. H., Nguyen Q. Uy., R. I. Mckay and P. M. Tuan, "Initializing PSO with Randomized Low-Discrepancy Sequences: The Comparative Results", In Proc. of IEEE Congress on Evolutionary Algorithms, 2007, pp. 1985 – 1992.

[8] K.E. Parsopoulos and M.N. Vrahatis, "Particle Swarm Optimization in noisy and continuously changing environments", in Proceedings of International Conference on Artificial Intelligence and soft computing, 2002, pp. 289-294.

[9] R. Brits and A.P. Engelbrecht and F. van den Bergh. A niching Particle Swarm Optimizater. In proceedings of the fourth Asia Pacific Conference on Simulated Evolution and learning, 2002, pp 692 – 696.

[10] R. Brits and A.P. Engelbrecht and F. van den Bergh.Solving systems of unconstrained Equations using Particle Swarm Optimization, In proceedings of the IEEE Conference on Systems, Man and Cybernetics, Vol. 3pp. 102 – 107, 2002.

[11] J. G. van der Corput, Verteilungsfunktionen. Proc. Ned. Akad. v. Wet., 38: pp. 813–821, 1935.

[12] J. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," Numerische Mathematik, Vol. 2, 1960, pp. 84 – 90.

[13] I. M. Sobol', "On the distribution of points in a cube and the approximate evaluation of integrals," USSR Computational Mathematics and Mathematical Physics, Vol. 7, 1967, pp. 86 – 112.

[14] I. M. Sobol', "Uniformly distributed sequences with an additional uniform property," USSR Computational Mathematics and Mathematical Physics, Vol. 16, 1976, pp. 236 – 242.

[15] J. Kennedy, "The Particle Swarm: Social Adaptation of Knowledge", IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ, 1997, pp. 303 – 308.

[16] R. C. Eberhart and Y. Shi, "Particle Swarm Optimization: developments, Applications and Resources", IEEE Int. Conference on Evolutionary Computation, 2001, pp. 81 – 86.

[17] Y. H. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer", IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 1998, pp. 69 – 73.

[18] H. M. Chi, P. Beerli, D. W. Evans, and M. Mascagni, "On the Scrambled Sobol Sequence", In Proc. of Workshop on Parellel Monte Carlo Algorithms for Diverse Applications in a Distributed Setting, LNCS 3516, Springer Verlog, 1999, pp. 775 – 782.

[19] Gehlhaar and Fogel, "Tuning Evolutionary programming for conformationally flexible molecular docking", In proceedings of the fifth Annual Conference on Evolutionary Programming, 1996, pp. 419 – 429.

[20] Y. Shi, R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", In Proc. of IEEE Congress on Evolutionary Computation, 1999, pp. 1945 – 1950.