Procedia Computer Science

Volume 80, 2016, Pages 772–781

ICCS 2016. The International Conference on Computational Science

Particle Swarm Optimization Simulation via Optimal Halton Sequences

Ganesha Weerasinghe[1], Hongmei Chi[2], and Yanzhao Cao[1]

[1] Department of Mathematics and Statistics, Auburn University, Auburn, Alabama, U.S.A.
{ ksw0013,yzc0009}@auburn.edu

[2] Department of Computer and Information Sciences, Florida A&M University, Florida, U.S.A.
hongmei.chi@famu.edu

Abstract

Inspired by the social behavior of the bird flocking or fish schooling, the particle swarm optimization (PSO) is a population based stochastic optimization method developed by Eberhart and Kennedy in 1995. It has been used across a wide range of applications. Faure, Halton and Vander Corput sequences have been used for initializing the swarm in PSO. Quasirandom(or low-discrepancy) sequences such as Faure, Halton, Vander Corput etc are deterministic and suffers from correlations between radical inverse functions with different bases used for different dimensions. In this paper, we investigate the effect of initializing the swarm with scrambled optimal Halton sequence, which is a randomized quasirandom sequence. This ensures that we still have the uniformity properties of quasirandom sequences while preserving the stochastic behavior for particles in the swarm. Numerical experiments are conducted with benchmark objective functions with high dimensions to verify the convergence and effectiveness of the proposed initialization of PSO.

Keywords: Randomized Low-discrepancy sequences, optimal Halton sequence, Particle Swarm Optimization, Stochastic optimization simulation

1 Introduction

The particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995. It is a optimization method inspired by the social behavior of the bird flocking or fish schooling [2]. PSO has been used across a wide range of applications. Areas where PSO have shown particular promise include multimodal problems and problems for which there is no specialized method available or all specialized methods give unsatisfactory results [11, 7, 8, 6].

Over the past two decades researches of PSO have been focused on two main aspects of PSO algorithm: initialization of particles and parameter selection. Initialization of particles plays an important role in population based optimization techniques. If the swarm population does

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.367

not cover the search area efficiently, it may not be able to locate the global optima and may converge to a local optimum point. Quasirandom sequences like Vander Corput sequence and Halton sequence provides better estimation in population search algorithms rather than Monte Carlo sequences [9],[10].

Quasirandom sequences are more evenly distributed over the D dimensional unit cube, thus it improves the accuracy of the estimation. But the drawback here is, that the the original Halton sequence suffers from correlation effect between radical inverse functions with different basses used for different dimensions [1], and also it provides deterministic behavior as opposed to the stochastic behavior. It is important to have stochastic initialization since PSO is a stochastic optimization method.

In this paper, we initialize the particles using randomized quasirandom sequence. This guarantees the uniformity properties of the quasirandom sequences and random permutation of the digits provides stochastic behavior.

The article is organized as follows. Section 2, Introduces the Standard PSO algorithm, In section 3, Optimal Halton sequence (Scrambled Halton sequence) is defined. Numerical results are discussed in Section 4. Finally Conclusion and Future work for this paper is presented in section 5.

2 The PSO Method

2.1 Standard PSO

A standard PSO algorithm maintains a population of M particles, and places them in the search space of the objective function. PSO defines each particle's position as a potential solution to the function to be optimized, and then searches for the optima by updating its position in every iterative step. Each particle is associated with a velocity which directs the flying of the particle toward a new, presumably better, position/solution. The particles fly through the problem space by following the current optimum particles. In every iteration, each particle's velocity is updated by following two "best" values. The first one is current best solution it has achieved so far. This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called *gbest*.

After finding the two best values, the particle updates its velocity and positions with following equations.

$$\vec{v}_{t+1} = w\vec{v_t} + c_1 r_{1,t+1}(\vec{pbest}_t - \vec{a_t}) + c_2 r_{2,t+1}(\vec{gbest}_t - \vec{a_t}). \tag{1}$$

$$\vec{a}_{t+1} = \vec{a_t} + \vec{v}_{t+1}. \tag{2}$$

Let D be the dimension of the search space, and let $a_t^{k,j}$ denote the k^{th} particle's j^{th} component at time t. Then $\vec{a^k}_t = (a_t^{k,1}, a_t^{k,2}, ..., a_t^{k,D})$ is the k^{th} particle's position at time t, $\vec{v^k}_t = (v_t^{k,1}, v_t^{k,2}, ..., v_t^{k,D})$ is the k^{th} particle's velocity at time t. $\overrightarrow{pbest_t^k} = (pbest_t^{k,1}, pbest_t^{k,2}, ..., pbest_t^{k,D})$ and $\overrightarrow{gbest_t} = (gbest_t^1, gbest_t^2, ..., gbest_t^D)$ are the vectors of current best values and global best values, respectively. $r_{1,t+1}$ and $r_{2,t+1}$ are uniformly distributed random numbers between 0 and 1. There are three factors w, c_1 and c_2 in equation (1). w factor is called the inertia weight and is simply a constant while c_1 and c_2 are the cognitive(or personal or local) weight and social or global weight respectively.

2.2 Algorithm

Without loss of generality, consider a minimization problem in the D dimensional space, where $f \in \mathbf{C}$, $\mathbf{C}$ is the set of bounded, continuous functions and $f : \Re^D \to \Re$.

$$\begin{gathered} Minimize : f(\vec{a}) \\ Subject\ to : \\ g_k(\vec{a}) \leq 0 \qquad k = 1, 2, ...p \\ h_m(\vec{a}) = 0 \qquad m = 1, 2, ...q \\ a^j_{min} \leq a^j \leq a^j_{max} \qquad j = 1, 2, ...D \end{gathered} \tag{3}$$

where $\vec{a} = (a^1, a^2, ..., a^D)$ and p, q are number of inequality and equality constraints respectively.

Algorithm of Standard PSO

1. Initialize a population array of M particles with random positions and velocities on D dimensions, in the search space.

   Loop

2. For each particle evaluate the Objective function in D variables.

3. Compare each particle's objective function value with its $\vec{pbest^k}_{t-1}$ value. If current value is better than $\vec{pbest^k}_{t-1}$, then set $\vec{pbest^k}_t$ equal to the current value. i.e

$$\vec{pbest^k}_t = \begin{cases} \vec{a^k}_t, & \text{if } f(\vec{a^k}_t) < f(\vec{pbest^k}_{t-1}). \\ \vec{pbest^k}_{t-1}, & \text{if } f(\vec{a^k}_t) \geq f(\vec{pbest^k}_{t-1}). \end{cases} \tag{4}$$
$$for\ k = 1, 2, ..., M.$$

4. Identify the particle in the swarm with the best success so far and assign its position to the variable $\vec{gbest}_t$.
   Choose $\vec{a^p}_t$ s.t.$f(\vec{a^p}_t) \leq f(\vec{a^k}_t)$ for all $k = 1, 2, ..., M$

$$\vec{gbest}_t = \begin{cases} \vec{a^p}_t, & \text{if } f(\vec{a^p}_t) < f(\vec{gbest}_{t-1}). \\ \vec{gbest}_{t-1}, & \text{if } f(\vec{a^p}_t) \geq f(\vec{gbest}_{t-1}). \end{cases} \tag{5}$$

5. Update each particle's j^{th} dimension of velocity according to the following equation, for $k = 1, 2, ..., M$ and $j = 1, 2, ..., D$

$$v^{k,j}_{t+1} = wv^{k,j}_t + c_1 r_{1,t+1}(pbest^{k,j}_t - a_t) + c_2 r_{2,t+1}(gbest^j_t - a_t). \tag{6}$$

   To ensure that each component of $\vec{v^k}_{t+1}$ is kept within the search space, make the following modification

$$v^{k,j}_{t+1} = \begin{cases} v^j_{min}, & \text{if } v^{k,j}_{t+1} < v^j_{min}. \\ v^{k,j}_{t+1}, & \text{if } v^j_{min} \leq v^{k,j}_{t+1} \leq v^j_{max} \\ v^j_{max} & \text{if } v^j_{max} < v^{k,j}_{t+1}. \end{cases} \tag{7}$$

   where v^j_{min} and v^j_{max} are determined from constraints of the objective function or by setting $v^j_{min} = a^j_{min}$ and $v^j_{max} = a^j_{max}$.

6. Update each particle's j^{th} dimension of position according to the following equation, for $k = 1, 2, ..., M$ and $j = 1, 2, ..., D$

$$a_{t+1}^{k,j} = a_t^{k,j} + v_{t+1}^{k,j}. \tag{8}$$

To ensure that each component of $\vec{a^k}_{t+1}$ is kept within the search space, make the following modification

$$a_{t+1}^{k,j} = \begin{cases} a_{min}^j, & \text{if } a_{t+1}^{k,j} < a_{min}^j. \\ a_{t+1}^{k,j}, & \text{if } a_{min}^j \leq a_{t+1}^{k,j} \leq a_{max}^j. \\ a_{max}^j & \text{if } a_{max}^j < a_{t+1}^{k,j}. \end{cases} \tag{9}$$

End of the Loop

7. If a criterion is met (after a certain number of iterations or until particle position converge to a certain value) exit loop.

3 Scrambled Halton Sequence

Unlike pseudorandom numbers, there are only a few common choices for quasirandom number generation. However, by scrambling a quasirandom sequence, one can produce a family of related quasirandom sequences. Finding one or a group of optimal quasirandom sequences within this family is an interesting problem, as such optimal quasirandom sequences can be quite useful for enhancing the performance of ordinary quasi-Monte Carlo. The process of finding such optimal quasirandom sequences is called the derandomization of a randomized (scrambled) family of quasirandom sequences. In addition to providing more quasirandom sequences for quasi-Monte Carlo applications, derandomization can help us to improve the accuracy of error estimation provided by randomized quasi-Monte Carlo. This is due to the fact that one can find a set of optimal sequences within a family of scrambled sequence family, and use sequences within this set for error estimation. In this section, we give a detailed description how to derive optimal Halton Sequences.

A classical family of low-discrepancy sequences are Halton sequences [4], which are bases on the radical inverse function defined as follows:

$$\phi_p(n) \equiv \frac{b_0}{p} + \frac{b_1}{p^2} + ... + \frac{b_m}{p^{m+1}}, \tag{10}$$

where p is a prime number and expansion of n in base b is given as $n = b_0 + b_1 p + ... + b_m p^m$, with integers $0 \leq b_j < p$.
Since Halton sequence X_n in $(0, 1]^s$ is defined as

$$X_n = (\phi_{p_1}(n), \phi_{p_2}(n), ..., \phi_{p_s}(n)), \tag{11}$$

where $p_1, p_2, ..., p_s$ are pairwise co-primes. In practice, we always use the first s primes as the bases.

Comparison to other low-discrepancy sequences, Halton sequences are easier to implement. However, a problem with Halton sequence comes from the correlations between the radical inverse functions for different dimensions. The correlations cause the Halton sequence to have poor 2-D projection for some pairing coordinates. In order to improve the quality of Halton sequence, the scrambled Halton sequence can break the cycle and correlation among dimensions.

Scrambled Halton sequence can help us to ignore the number of points and obtain good quality of Halton sequence.

By analyzing the inner property of points in each coordinate, correlations are related to the most significant bit. We permute the most significant bits of each Halton point according to coordinate. The period of points in each coordinate is the base. Permutation of most significant bit of each point is the same as permutation in $\{\phi_p(1), \phi_p(2), \ldots, \phi_p(b)\}, \{\phi_p(b+1), \phi_p(b+2), \ldots, \phi_p(2b)\}, \ldots$. The advantage of this procedure is that we have the same code as the original Halton sequence. The only thing we need to do is to permute the points according to each coordinate, and output the scrambled Halton sequence. This permutation of Halton sequence does not change its uniformity in one dimension and just change the position of one point [1]. In this practice, we followed recommendation from early implementation[3] to skip certain number of n instead $n = 1$, we dropped first 5000 points and start with $n = 5001$. This has no effect on asymptotic performance, but it dramatically improves practical performance when dimension is high. It is always safe to skip a couple thousand points when we are using quasirandom sequences.

4 Numerical Results

To implement the PSO algorithm, the values of the parameters $\{\omega, c_1, c_2\}$ are needed. According to Ming Jiang et al [5], suggested parameter tuple in literature are $\omega = 0.729, c_1 = 2.8\omega, c_2 = 1.38\omega$ or $\omega = 0.729, c_1 = c_2 = 1.49$ or $\omega = 0.6, c_1 = c_2 = 1.7$. Among these three, $\omega = 0.729, c_1 = c_2 = 1.49$ is the most widely used parameter tuple, so we choose that to perform our numerical results. Stopping criteria for the algorithm is taken as the maximum number of iterations. Millie Pant et al [10] have used benchmark problems to analyse the performance of PSO, if particles are initialized with Vander-Corput sequence. We have chosen the same functions to compare the optimality of the solution, which are summarized in table 1. The optimum point of each function is at 0. In order to have a fair comparison , we have set number of particles to 20 and 40. Also we have our comparison in the same ranges.

Particles are initialized using 1) Halton sequence with base 3 for each direction , 2) Scrambled Halton sequence with base 3 for each direction and 3) dropping the first 5000 points of Scrambled Halton sequence. "Halton" denotes the result for initializing using Halton sequence with base 3, "Scrambled H." denotes the result for initializing using Scrambled Halton sequence with base 3 and "Drop 5000" denotes result for initializing after dropping first 5000 points of Scrambled Halton sequence with base 3. Since the initialization of the particles follows a random behavior each test is executed 30 times and took the mean of the values as the optimum value.

The results are summarized in tables 2-5. Here M denotes the number of particles and D denotes the dimension of the function. Millie Pant et al [10] , have performed the algorithm for 1000, 1500 and 2000 iterations. But since our value converge to the optimum point at 50 iterations, we performed the results only for 50 iterations. Also we checked the convergence for high dimensions i.e 70 and 100. Figures 1-4 show the convergence of global best value in the swarm for each function at 100 dimensions evaluated with 40 particles, in the range 1, and the one on right is an enlarged view of the same graph showing the convergence for iteration number 25 to 50.

Objective function	Range 1	Range 2
$f_1 = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^D$	$[-100, 100]^D$
$f_2 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} cos(\frac{x_i}{\sqrt{i+1}}) + 1$	$[-600, 600]^D$	$[-1000, 1000]^D$
$f_3 = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30, 30]^D$	$[-100, 100]^D$
$f_4 = 20 + e - 20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i))$	$[-32, 32]^D$	$[-100, 100]^D$

Table 1: Optimization test functions and it's Ranges

M	D	R1:[-5.12, 5.12]			R2:[-100, 100]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	4.5031e-05	9.4292e-05	1.4199e-05	0.0183	0.0047	0.0093
	20	6.0832e-05	1.7520e-04	9.1973e-06	0.0201	0.0047	0.0070
	30	6.9443e-06	2.6703e-04	1.3455e-04	0.0199	0.0226	0.0153
	50	5.7875e-04	5.5034e-05	1.5749e-05	0.6320	0.1052	0.0104
	70	8.0016e-05	8.1385e-05	7.6740e-05	0.0084	0.0428	0.0163
	100	3.7532e-04	5.7959e-05	1.8847e-04	0.1467	0.0216	0.0068
40	10	3.6110e-06	1.6048e-06	3.6206e-06	0.0040	0.0073	0.0051
	20	6.9260e-05	4.3515e-05	8.5975e-06	0.0059	0.0028	0.0079
	30	1.8150e-05	4.9339e-05	1.4129e-05	0.0058	0.0010	0.0240
	50	6.1877e-06	1.6712e-05	2.3661e-05	0.0015	0.0564	0.0022
	70	1.0746e-04	9.1970e-06	8.7913e-05	0.0032	0.0072	0.0044
	100	1.3386e-04	3.1211e-04	2.5485e-05	0.0043	0.0150	0.0181

Table 2: Results of test function f_1

M	D	R1:[-600, 600]			R2:[-1000, 1000]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	4.9462e-05	2.2319e-04	1.5571e-04	0.0014	7.6142e-04	1.6756e-04
	20	2.9423e-04	3.2445e-04	1.6124e-04	0.0040	6.5461e-04	5.4820e-04
	30	2.0535e-04	2.0806e-05	5.2344e-05	0.0016	8.8793e-04	2.1003e-04
	50	1.5963e-04	1.7287e-04	2.0185e-04	0.0032	6.7706e-04	7.7382e-04
	70	1.1883e-04	9.1621e-04	2.6643e-04	0.0018	9.9320e-05	0.0029
	100	1.4924e-04	3.6559e-04	8.6489e-04	0.0010	3.5493e-04	0.0022
40	10	1.1053e-05	2.0025e-05	8.0946e-06	3.4932e-04	2.4241e-04	4.9436e-05
	20	2.8379e-05	1.3041e-05	4.0735e-05	1.3493e-04	7.8787e-05	1.0553e-04
	30	2.9824e-04	5.4652e-04	7.1948e-04	6.9671e-05	5.0725e-05	8.3411e-04
	50	1.6720e-04	5.3928e-05	8.3615e-05	7.7936e-04	1.1867e-04	9.9799e-05
	70	3.1517e-04	3.2714e-05	3.6104e-05	5.2766e-04	4.2098e-04	1.4845e-04
	100	1.5062e-04	5.5221e-05	9.3912e-05	1.4093e-04	6.6336e-04	3.1576e-04

Table 3: Results of test function f_2

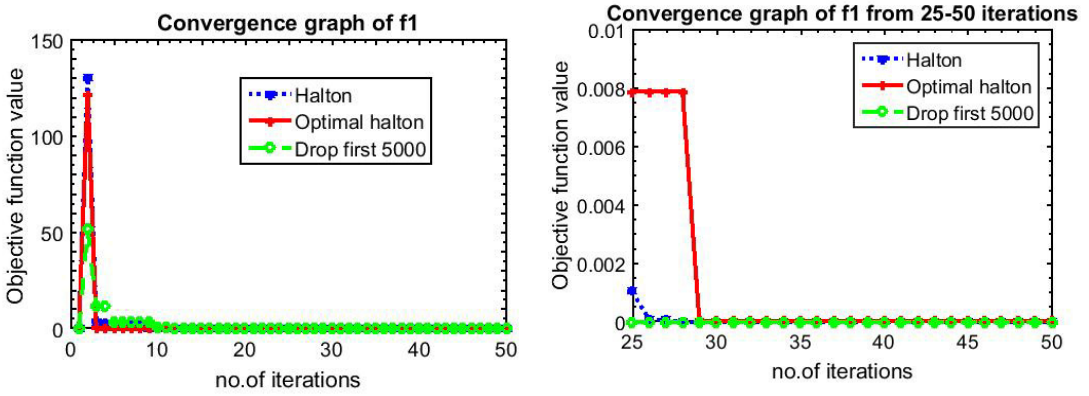

Figure 1: Convergence graph for function f1

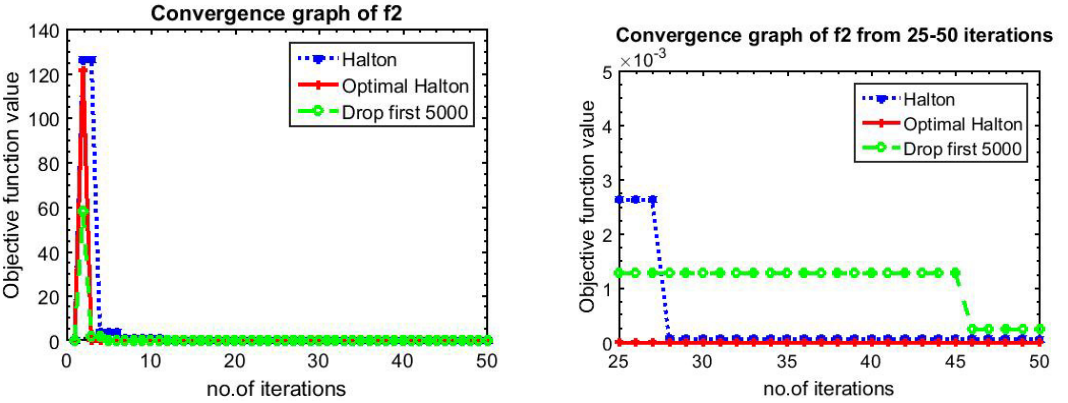

Figure 2: Convergence graph for function f2

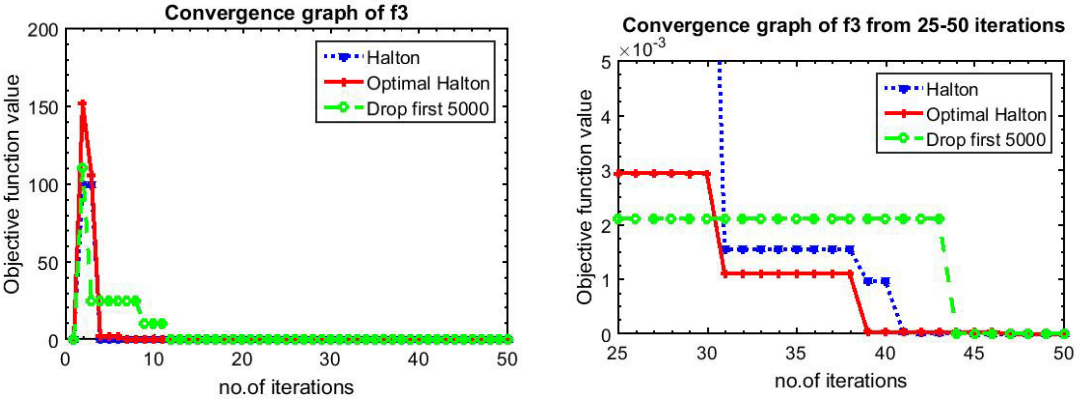

Figure 3: Convergence graph for function f3

M	D	R1:[-30, 30]			R2:[-100, 100]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	0.0342	0.0038	0.0023	0.0951	0.0048	0.1557
	20	0.0148	0.0011	0.0047	0.0090	0.1181	0.0222
	30	0.0051	0.0036	0.0021	0.0157	0.0383	0.0076
	50	0.0346	0.0013	4.2693e-04	0.0418	0.1161	0.0031
	70	0.0034	0.0427	0.0025	0.0527	0.0941	0.0272
	100	0.0018	0.0312	0.0087	0.1361	0.0305	0.2750
40	10	3.1437e-04	6.1063e-04	1.9113e-04	0.0134	0.0038	9.7550e-04
	20	7.4913e-04	1.8756e-04	1.7657e-04	0.0263	0.0070	0.0057
	30	7.2197e-04	1.2019e-04	3.9049e-04	0.0584	0.0077	0.0225
	50	0.0049	7.9631e-04	2.1287e-04	0.0257	0.0030	0.0017
	70	0.0031	6.0691e-04	0.0056	0.1823	0.0567	0.0495
	100	0.0026	7.2791e-04	0.0032	0.0412	0.0539	0.0209

Table 4: Results of test function f_3

M	D	R1:[-32,32]			R2:[-100, 100]		
		Halton	Scrambled H.	Drop 5000	Halton	Scrambled H.	Drop 5000
20	10	0.0014	0.0072	8.7490e-04	0.0035	0.0060	0.0194
	20	0.0025	0.0026	8.0518e-04	0.0067	0.0028	0.0057
	30	0.0021	9.5752e-04	0.0072	0.0044	0.0018	0.0039
	50	0.0032	0.0022	0.00241	0.0186	0.0062	0.0041
	70	0.0022	0.0017	0.0011	0.0028	0.0026	0.0062
	100	0.0025	5.9831e-04	4.4743e-04	0.0033	0.0033	0.0019
40	10	0.0010	7.9967e-04	5.4900e-04	0.0018	8.6368e-04	
	20	7.4003e-04	5.0134e-04	3.8245e-04	0.0041	0.0025	0.0022
	30	8.6885e-04	6.2990e-04	0.0015	6.2206e-04	0.0043	0.0018
	50	8.8984e-04	0.0010	0.0011	0.0038	0.0016	0.0029
	70	0.0016	5.4819e-04	4.3998e-04	0.0025	0.0025	0.0016
	100	0.0015	7.1370e-04	6.2701e-04	0.0044	0.0012	7.1727e-04

Table 5: Results of test function f_4

5 Conclusion and Future Work

In PSO, Particles' velocities are updated according to a random manner. Hence the particle's positions are random vectors. So it is important to initialize particles using random sequences. In this paper we showed the importance of using Randomized quasirandom sequences in initializing the particles. Scrambled optimal Halton sequence is one such sequence and we used that sequence for numerical experiments.

The Numerical results shows that for f_1 , f_3 and f_4 all three sequences provides similar results. For f_2 , Scrambled Halton and drop 5000 sequences provides better estimation than Halton sequence when the range is expanded and the number of particles is less.

From this analysis we conclude that , even though all three sequences provides similar results, Scrambled Halton sequence and Drop 5000 sequences are better in initializing the swarm due

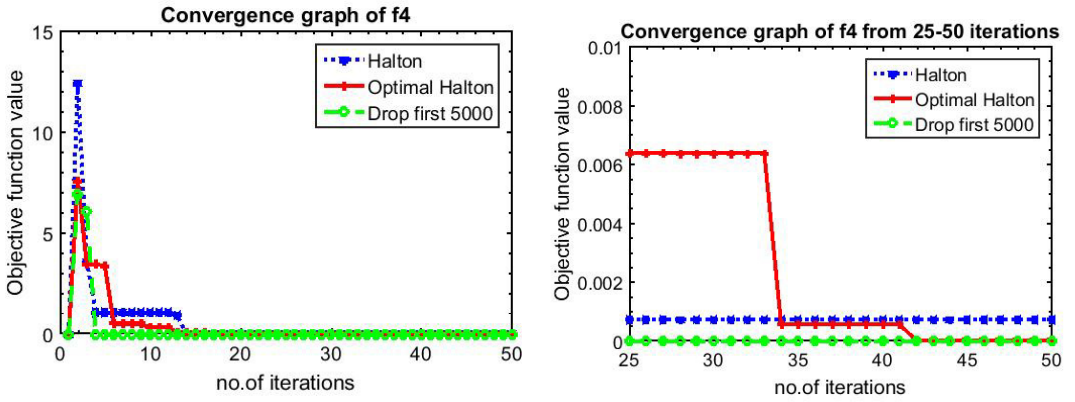

Figure 4: Convergence graph for function f4

to their random behavior. Also it is clear that accuracy can be improved by increasing the number of particles in the swarm, since it covers the search space more efficiently.

In future studies we plan to analyze convergence of particle's positions theoretically if we initialize particles using randomized quasirandom sequences.

References

[1] Hongmei Chi, Michael Mascagni, and T Warnock. On the optimal halton sequence. *Mathematics and computers in simulation*, 70(1):9–21, 2005.

[2] Russ C Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[3] Bennett L Fox. Algorithm 647: Implementation and relative efficiency of quasirandom sequence generators. *ACM Transactions on Mathematical Software (TOMS)*, 12(4):362–376, 1986.

[4] John H Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.

[5] Ming Jiang, YP Luo, and SY Yang. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, 102(1):8–16, 2007.

[6] A Rezaee Jordehi. Enhanced leader pso (elpso): a new pso variant for solving global optimisation problems. *Applied Soft Computing*, 26:401–417, 2015.

[7] Jing J Liang, A Kai Qin, Ponnuthurai Nagaratnam Suganthan, and S Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, 10(3):281–295, 2006.

[8] Yanmin Liu, Zhuanzhou Zhang, Yuanfeng Luo, and Xiangbiao Wu. An improved pso for multimodal complex problem. In *Intelligent Computing in Bioinformatics*, pages 371–378. Springer, 2014.

[9] Millie Pant, Radha Thangaraj, and Ajith Abraham. Low discrepancy initialized particle swarm optimization for solving constrained optimization problems. *Fundamenta Informaticae*, 95(4):511, 2009.

[10] Millie Pant, Radha Thangaraj, Crina Grosan, and Ajith Abraham. Improved particle swarm optimization with low-discrepancy sequences. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3011–3018. IEEE, 2008.

[11] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.