Dispersion-Based Population Initialization

Ronald W. Morrison

Mitretek Systems, Inc.
3150 Fairview Park Drive South
Falls Church, VA 22043-4519
ronald.morrison@mitretek.org

Abstract. Reliable execution and analysis of an evolutionary algorithm (EA) normally requires many runs to provide reasonable assurance that stochastic effects have been properly considered. One of the first stochastic influences on the behavior of an EA is the population initialization. This has been recognized as a potentially serious problem to the performance of EAs but little progress has been made in improving the situation. Using a better population initialization algorithm would not be expected to improve the many-run average performance of an EA, but instead, it would be expected to reduce the variance of the results, without loss of average performance. This would provide researchers the opportunity to reliably examine their experimental results while requiring fewer EA runs for an appropriate statistical sample. This paper uses recent advances in the measurement and control of a population's dispersion in a search space to present a novel algorithm for better population initialization. Experimental verification of the usefulness of the new technique is provided.

1 Introduction

Execution and analysis of an evolutionary algorithm (EA) normally requires many runs to provide reasonable assurance that any negative effects of merely "bad luck" in the stochastic processes have been overcome. One of the first stochastic influences on the behavior of an EA is the population initialization. This has been recognized as a potentially serious problem to the performance of EAs in [1], and, to a lesser extent, in [2], but little progress has been made in improving the situation. In the few cases where this situation is addressed at all, populations, in very low dimensionality problems, have been initialized using techniques like the Latin Hypercube [2]. The Latin Hypercube technique guarantees uniform placement along each axis, but uniform placement along individual axes does not ensure any level of uniformity throughout the search space.

EAs are generally executed many times to overcome stochastic anomalies, so using a better population initialization algorithm would not be expected to improve the average performance of an EA. Instead, it would be expected to reduce the variance of the results, without loss of average performance. This

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 1210–1221, 2003.

would provide researchers the opportunity to reliably examine their experimental results while requiring fewer EA runs for an appropriate statistical sample.

This paper provides a description of the population initialization problem, an introduction to low-discrepancy sequences, the derivation of a new and computationally efficient population initialization algorithm, and experimental results that demonstrate the usefulness of the algorithm.

2 Problem Description

We desire to improve the search capability of the initial population by spreading the individual members throughout the search space so that their positioning maximizes the probability of detection of a landscape feature of interest. Let's take a deeper look at more precisely what this means.

In the general case, for any specific population size, P, in N dimensions, this is equivalent to solving an N-dimensional sphere packing problem. The problem, however, is expressed in terms of the packing space size and the number of spheres, and must be solved for the sphere radius and the coordinates of the packed spheres. The sphere center coordinates would then be the population positions. While much research has been done in sphere packing problems [3] [4], efficient methods for solving the above problem have not been identified. Note that we would also like to avoid adding significant computational complexity to our EA.

The population placement problem becomes even more difficult if we wish to use a variety of population sizes. We would like the ability to initialize different sized population for a problem without needing to re-perform a complex population initialization algorithm. We would like, therefore, to be able to appropriately add or delete population members to an initialization distribution that we have already computed.

The identification of the largest unoccupied volume for the placement of the "next" population member in N-space already occupied by a population is an interesting problem. The normal N-dimensional space partitioning techniques such Delaunay triangulation [5], or other space partitioning methods involving polytopes, do not assist in the identification of the largest unoccupied N-dimensional volume. Application of these techniques to our problem could result in an unintentional division of the largest unoccupied volume by inappropriate space partitioning edge placement, rather than point placement.

Another area of research that could be used for placing points in N-dimensional space includes meshing techniques such as t-m-s nets [6]. As a powerful mechanism for placing points uniformly in N-dimensional space, t-m-s nets cover space in base b in dimension N, requiring b^N points to assure uniform spatial coverage. Since the process starts by partitioning the space into b^N cells and we wish to use a varying number of points, the mesh-generating techniques are generally inappropriate for our application.

Since we are trying to place all population members such that the distance to a any point in the search space is minimized, ideally we would like to place an

additional population member into an existing population at the mid-point of the largest unoccupied symmetrical volume. What we are in need of is a computationally efficient algorithm that is better than a random number generator in efficiently placing a population uniformly throughout a search space, occupying unoccupied volumes first. The specialty field in mathematics that has done some exploration into this type of problem is called low-discrepancy sequence generation [7]. The key idea in this field of mathematics is discrepancy measurement, so first we will provide a short introduction to discrepancy measurement.

3 Discrepancy Measurement

Discrepancy measurement identifies how uniformly a set of points sample a search space. Discrepancy measurement involves concepts that are loosely based on the same ideas as the Kolomogorov-Smirnov (K-S) test, when used to compare a distribution of points to the uniform distribution. The K-S test compares an ordered set of P points X_i to a continuous distribution by computing the difference between the discrete cumulative distribution of the points with cumulative distribution of the desired continuous distribution. The hypothesis regarding the distributed form of the points is rejected if a test statistic, D, is greater than the critical value.

Unfortunately, the cumulative probability distribution is not well defined in more than one dimension. So for higher dimensions the common measure of spatial uniformity is discrepancy. While not providing a test statistic for uniformity, discrepancy can be used to compare sequences of points in N-space to determine which sequence is more uniform. Discrepancy measures consider the N-dimensional $\Re^N$ space modulo 1 or, equivalently, the N-dimensional torus $T^N = \frac{\Re^N}{\mathcal{Z}^N}$. Instead of identifying the largest difference between cumulative distributions of points and the desired distribution, as the K-S test does, discrepancy identifies the largest difference between the proportion of points included in an N-dimensional space partition and the proportion of the total volume (Lebesgue measure) included in that partition, over all possible space partitions.

Discrepancy, $\mathbb{D}_P$, of a sequence of points, $(x_p)_{p\geq 1}, x_p \subseteq \Re^N$, as further elaborated in [7], is useful for comparing the uniformity sequences, because a sequence is uniformly distributed modulo 1 if and only if:

$$\lim_{P\to\infty} \mathbb{D}_P(x_P) = 0. \tag{1}$$

As a mathematically sound method for computing the uniformity of a distribution of points in an N-dimensional space, it is unfortunate that discrepancy measurement is generally too computationally expensive to be practically employed in most EA problems involving assessment of the dispersion of points in a search space. Recently [8] developed a computationally efficient method for measuring a population's dispersion throughout a search space, called the dispersion index, Δ. This measure is based on on the concepts of discrepancy theory and provides practical improvements over the more commonly used diversity measurements.

Applying discrepancy theory concepts to our population initialization problem, we turn to research in the generation of low-discrepancy sequences. This research is focused on methods for placing points uniformly in multi-dimensional spaces. Most of the research into low-discrepancy sequences is involved in establishing upper and lower bounds on the discrepancy of various sequence generation algorithms, through algorithmic proofs or example construction.

The research in this field yields sequences with the smallest values of discrepancy (in the limit) currently known [9] [10]. There is, however, no evidence that these sequences will perform well in an application such as ours, where only a small number of points near the beginning of the sequence will actually be used. Additionally, our application has computational efficiency requirements, and none of the research appears to adequately address this requirement. In the following sections, applying the low-discrepancy sequence generation methods, we derive a computationally efficient, heuristic sequence generation algorithm for an N-dimensional space that has the attributes we desire.

3.1 Population Placement Basis

For the construction of our population placement algorithm, we used some of the research on the $k\alpha$ sequence for an irrational α. This is a well-studied sequence and is known to be uniformly distributed modulo 1 [7].

This sequence has been extended into N-dimensions, through what is known as the Kroenecker sequence [10]. The Kroenecker sequence is based on the following theorem, from [7]. Let $\beta_1, ..., \beta_N \in \Re$. Then:

Theorem 1:
The N-dimensional sequence: $x_k = (k\beta_1, ...k\beta_N)$ is uniformly distributed modulo 1 if and only if $1, \beta_1, ..., \beta_N$ are linearly independent over the integers $\mathcal{Z}$.

This is an extension of the $k\alpha$ sequence since, for $\beta 1 ... \beta N$ to be linearly independent over the integers, there are no integers $m_i \in \mathcal{Z}$ such that $1, m_1\beta 1 + ... + m_N\beta_N = 0$. In other words, the β's must be irrational. So, using our notation convention, we will refer to these numbers in future references as α's (i.e., $k\alpha_1, ...k\alpha_N$).

The mathematical study of low-discrepancy sequences is usually focused on the sequence discrepancy in the limit for large numbers of points. We, however, aren't really interested in theoretical convergence for large sequences. We are interested in the uniformity of the distribution of the first few points (perhaps as high as several hundred) of the sequence. The extreme computational efficiency of the Kroenecker sequence, once the α's are identified, makes it very attractive for our application.

So our contribution here is to devise a set of rules for using the concepts of the Kroenecker sequence to construct a reasonably uniformly distributed set of points in N-space for the range of dimensions and range of number of points of interest for EA applications. We will then use those rules to derive a set of α's that EA practitioners and researchers can use. To understand the problem, we

will first illustrate that the Kroenecker sequence does not, by itself, necessarily provide the desired uniformity of the initial points in a sequence. Figure 1 illustrates one type of non-uniform initial point distributions that can be generated a by badly chosen, but irrational, α's in two dimensions. This sequence will be uniformly distributed in the long run, but is a very bad choice with respect to the initial few points.

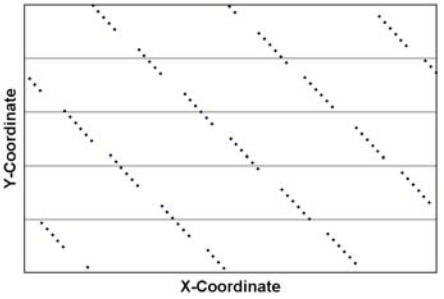

Fig. 1. 50 Kroenecker Sequence Points #1

4 Heuristic Rules

What we need is "good mixing" for the first few hundred points of the sequence. To accomplish this, we have developed a set of heuristic rules to choose α's with good inter-dimensional mixing. The rules will first be delineated, then the reasons for each of these heuristics will be described.

1. Base the incremental intervals for each dimension on multiples of the "Golden Ratio" which is: $\phi = \frac{\sqrt{5}+1}{2}$.
2. Select the multiples of ϕ for each dimension from sequences of prime numbers.
3. Examine the resultant modulo 1 sequences for the quasi-period with which they revisit a value near the initial value (this is done by examining the sequential distance difference as described below). If a multiplier results in either a very short or trivial sequence of sequential differences, or closely matches the sequential difference sequence of a previously selected multiplier, do not use that multiplier, use the next prime number instead.

The reasons for these heuristics are fairly straightforward. The first rule is based on the fact that to achieve good mixing in the initial points of the sequence, we need a "very" irrational number. Irrational numbers are classified by how easy they are to approximate with continued-fraction ratios of integers. For example, $\sqrt{2}$, which is irrational, would be inappropriate to use since it is well approximated by a continued fraction sequence.

The "Golden Ratio" mentioned previously, normally represented by ϕ, is the irrational number that is most poorly approximated through continued fractions [11]. This is the obvious choice for uniformly of a one-dimensional placement, since the more irrational α is, the more uniformly distributed is the first part of the sequence of P points $P_i = \{K_i\alpha\}$, where $K_i = 1, 2, 3...P$ [10]. However, the use of ϕ only guarantees best placement in one dimension. If the same α (or any integer multiple of the same α) is used for a Kroenecker sequence in more than one dimension, the points are merely placed along a diagonal in those dimensions. This is why the Kroenecker sequence requires linear independence of the α's among $\mathcal{Z}$. Since it can be very difficult to prove that a number is irrational or that multiple irrational numbers are linearly independent over $\mathcal{Z}$, this brings us to the second heuristic rule.

The second rule is to select the multiples of ϕ for each dimension from sequences of prime numbers, avoiding the prime numbers 1 and 2. What we need to accomplish here is to force the pair-wise relationship between dimensions to be irrational. Ratios of prime numbers approximate irrational numbers and this method makes the ratios of the step sizes between any two dimensions relatively irrational. [1] This approximation is adequate for our purposes, as long as the sequences generated provide good placement and good inter-dimensional mixing. Since, however, multiples of ϕ do not necessarily have the same placement uniformity as ϕ, we come to the third heuristic rule.

This third heuristic rule is designed to improve mixing across multiple dimensions. In a sequence $kZ_1\alpha$ modulo 1, for $k = 1, 2, 3, ...$ and $Z_i \in \mathcal{Z}$, values within a distance ε of the first point are visited quasi-periodically, with the sequential differences between these quasi-periods forming an easily recognizable pattern, but with occasional and irregular breaks in the pattern. An example of this is shown as Figure 2. In this figure, the distance to the starting point of a Kroenecker sequence is plotted for 50 points. As can be seen, for $\varepsilon = .10$ the distance is less than ε at $k = 6, 14, 18, 22, 26, 31, 35, 39, 43, 48....$ The sequential difference between the k values are $8, 4, 4, 4, 5, 4, 4, 4, 4, ...$ When these sequential differences form trivial patterns like this (or, for example, $3, 3, 3, 3, 3, 3, 5, 3, 3, 3, 3, 3, 3, 5$), or are identical to the patterns already selected for other prime multipliers, the fill patterns across multiple dimensions are usually not adequately mixed. These sequences are easily checked using a simple spreadsheet, and, if an undesired sequence is encountered, the next prime number in the sequence should be selected as the multiplier.

Table 1 provides recommended α values for 1 through 12 dimensions that were derived using the heuristic rules described above. The Table also provides the Dispersion Index, Δ from [8] for populations of the first 50 points generated using these α's. The Dispersion Index, Δ measures the uniformity of the population dispersion and has a maximum value of 1.00 for a completely uniform distribution. While the recommended α's in Table 2 are not guaranteed to be

[1] Sequential very large prime numbers must also not be used, since the ratio of two sequential very large prime numbers closely approximates 1.0 and will result in a violation of the next heuristic rule.

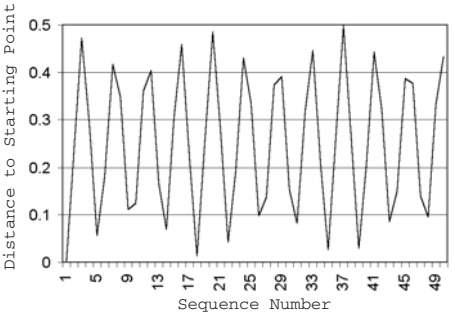

Fig. 2. Quasi-Periodic Distance to the First Point

the optimal α's for use with any specific EA problem, these prime numbers are known to provide reasonably uniform spatial distributions for the first 200 points in dimensions up through 12.

Table 1. Prime α's and Dispersion Indices for 50 Points, Dimensions 1 through 12

Dimension	α	Δ
1	41	0.9871
2	43	0.9841
3	47	0.9702
4	59	0.9874
5	83	0.9738
6	107	0.9795
7	109	0.9853
8	173	0.9853
9	311	0.9853
10	373	0.9910
11	401	0.9999
12	409	0.9910

The resultant heuristic sentinel placement algorithm works as follows. First scale and offset all search-space dimensions as necessary to set each dimensional range from 0 to 1. Second, randomly place the initial sentinel point, x_i, within the search space, for i equal 1 to N. Finally, compute the coordinates of each subsequent point P, $x_{i,p} \in P$, as:

$$x_{i,p} = mod1[x_{i,p-1} + Z_i\phi], \tag{2}$$

where the Z_i's are selected for each dimension in accordance with the heuristic rules specified.

As can be seen, the coordinates of each point can easily (and very computationally efficiently) be determined from the location of the previous point, once

the Z_i's are selected through an off-line process for the range of dimensions and population sizes appropriate for your problem domain. Note that nothing more need be known about the problem space than the number of dimensions and the approximate size of the population of sentinels to be used, and that this information is used off-line in a simple spreadsheet calculation.

4.1 Population Placement in Very High-Dimensional Search Spaces

Very high-dimension search spaces would require a large set of prime numbers that obey the heuristic rules for selection. These can be difficult to find, although all effort expended in finding them is performed off line and does not affect the EA performance. More important, however, is the fact that it is probably not worth the effort. In large dimensional spaces, the distances between points increase rapidly. With very large distances between only a few points, the benefit of placing population members "uniformly" versus "randomly" diminishes. For this and other reasons, several mathematical researchers have expressed doubts about the usefulness of uniform distribution methods for dimensions higher than about 12 [12] [9]. Because of this, if the dimensionality of the problem is greater than 12, random population placement is recommended. Up through dimension 12, Table 1 provides a pre-computed set of α's.

4.2 Population Placement in Complex Search Spaces

Up until this point, we have only been describing population placement in symmetric, real-numbered search spaces. Fortunately, the transition to other search spaces is quite straightforward.

The use of this technique for asymmetric search spaces is straightforward. Since the population placement algorithm is so computationally efficient, simply compute the population as if the search space was symmetric in all axes (using the largest axis, scaled 0 to 1 as the basis), and disregard any points falling outside the search space. Compute additional points, continuing to discard those falling outside of the search space, until the desired number of population members fall within the search space.

Similarly, the use of this technique for search spaces involving non-binary combinatorial problems is also straightforward. If, for example, one of the axes of the search space involves four non-ordinal values, a simple partitioning of the axis into four equal-length segments to represent the four values will suffice for problem initialization. Since the placement algorithm assures reasonably uniform inter-dimensional dispersion for problem initialization, this technique will provide an appropriate representation of population members dispersed throughout the search space.

5 Experimental Verification

To examine the usefulness of the sentinel placement algorithm for population initialization, a simple genetic algorithm (GA) with gray code binary represen-

tation, fitness-proportional selection, uniform crossover, and a mutation rate of 0.001 was used. This GA was run against a variety of different problems of varying complexity. A few representative results are provided here. The problems presented are:

- De Jong Test Function 2, also called Rosenbrock's Function, in 2 dimensions [13].
- De Jong Test Function 3, a step function, in 5 dimensions [13].
- Michalewicz's Function, in 10 dimensions [13].

A population of 25 was used. The De Jong test functions #2 and #3 and the Michalewicz test function are usually implemented as minimization functions, but were implemented as maximization functions for these experiments, applying the following equations respectively:

$$f(\overline{x}) = 3907 - 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \text{ for } x_{1,2} \in [-2.048, 2.048]. \quad (3)$$

$$f(\overline{x}) = 25.0 - \sum_{i=1}^{N} |x_i|, \text{ for } x_i \in [-2.048, 2.048]. \quad (4)$$

$$f(\overline{x}) = \sum_{i=1}^{N} ((\sin(x_i))(\sin(\frac{ix_i^2}{\pi}))^{20}), \text{ for } x_i \in [0, \pi]. \quad (5)$$

The GA was run on each of these problems 100 times using random population initialization and 100 times using placed population initialization (recall that, since the placement algorithm uses a random place for the first placed point, each "placed population" will be different, but equivalently dispersed in the search space). Since these are static problems, the average "best-so-far," along with its variance, for the 100 runs are reported as the measures of effectiveness in this chapter.

The results of the first 25 generations of the above experiments are graphed in Figures 3 through 8. As can be seen in Figures 3 and 4 there appears to be a considerable reduction in variance for De Jong Test Function 2. The other charts show less dramatic results, so the results were further analyzed to determine their statistical significance.

6 Analysis

At any specific generation, the F-statistic can be used to determine whether the differences between the two variances are statistically significant.

Table 2 provides a look at the significance of the variance difference seen at generation 10 in each of the figures from the previous section. For each technique, the mean best-so-far (BSF) fitness at generation 10, the variance of the mean

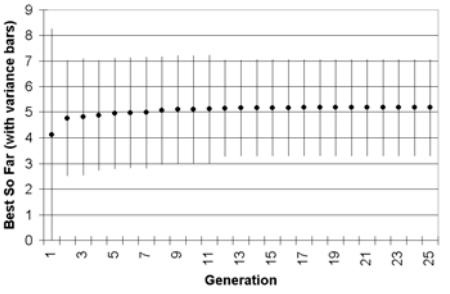

Fig. 3. Best So Far, Random Initialization, DeJong Test Function 2

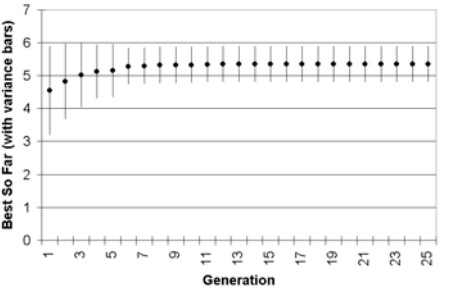

Fig. 4. Best So Far, Placed Initialization, DeJong Test Function 2

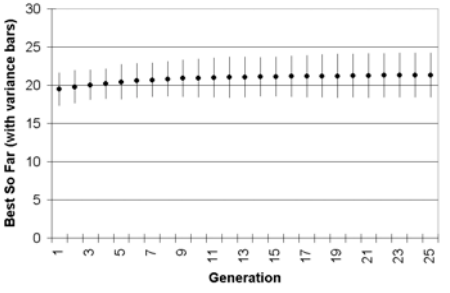

Fig. 5. Best So Far, Random Initialization, DeJong Test Function 3, 5-Dimensions

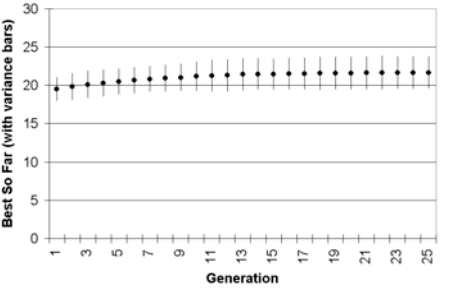

Fig. 6. Best So Far, Placed Initialization, DeJong Test Function 3, 5-Dimensions

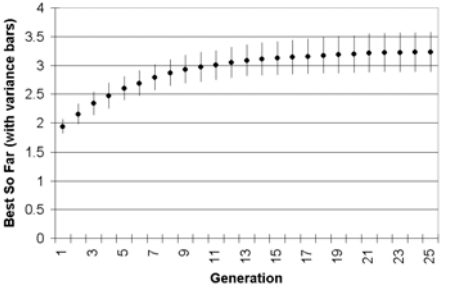

Fig. 7. Best So Far, Random Initialization, Michaeliwicz's Function, 10-Dimensions

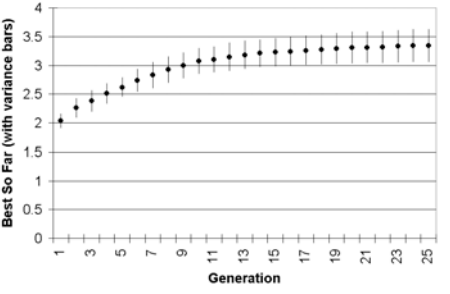

Fig. 8. Best So Far, Placed Initialization, Michaeliwicz's Function, 10-Dimensions

Table 2. Results of Different Population Initialization at Generation 10

Test	BSF Mean	Variance	F	Confidence
De Jong 2 2D random	3905.10	2.12	3.867	99+%
De Jong 2 2D placed	3905.32	0.55	-	-
-	-	-	-	-
De Jong 3 5D random	20.90	2.53	1.326	76%
De Jong 3 5D placed	21.15	1.91	-	-
-	-	-	-	-
Michalewicz 10D random	3.00	0.26	1.135	77%
Michalewicz 10D placed	3.10	0.23	-	-

fitness at generation 10, the F statistic computed as the ratio of the squares of the two variances, and the confidence level that the variances are different.

As expected, there was no significant difference in the mean best-so-far performance of the EAs attributable to the population initialization algorithm (the t statistic is therefore omitted for brevity), but there was generally a reduction in the variance of the mean best-so-far performance. As addressed previously, at high dimensions the advantage of the placement algorithm is reduced.

7 Summary

In this paper we have derived a population initialization algorithm for EAs in static environments. Use of the technique does not improve mean best-so-far performance over a large number of runs. Instead, it reduces the variance of the mean best-so-far performance without loss of average performance, thereby providing researchers the opportunity to reliably examine their experimental results needing fewer EA runs for an appropriate statistical sample. This may provide an opportunity for researchers to address more complex problems without an attendant increase in required computational resources.

References

1. Kallel, L. and Schoenauer, M.: Alernative Random Initialization in Genetic Algorithms. In: Proceedings of the Seventh International Conference on Genetic Algorithms. Morgan Kaufman (1997) 268–275
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers (2002)
3. Shimada, K. and Gossard, D.: Bubble Mesh: Automated Triangular Meshing of Non-manifold Geometry by Sphere Packing. In: Proceedings of the Third Symposium on Solid Modeling and Applications. IEEE (1995) 225–237
4. Rush, J.: Sphere Packing and Coding Theory. In: Handbook of Discrete and Computational Geometry. CRC Press (1997) 185–208
5. Weisstein, E.: Hypersphere. In: World of Mathematics. Wolfram Research, Inc. http://mathworld.wolfram.com (2001)

6. Neiderreiter, H.: Low-discrepancy and Low-dispersion Sequences. In: Journal of Number Theory 30-1 (1987) 51–70.
7. Drmota, M. and Tichy, R.: Sequences, Discrepancies and Applications, Lecture Notes in Mathematics 1651. Springer-Verlag (1997)
8. Morrison R.: Designing Evolutionary Algorithms for Dynamic Environments. Ph.D. Dissertation. George Mason University (2002)
9. Bratley, P., Fox, B. and Neiderreiter, H.: Implementation and Tests of Low-discrepancy Sequnces. In: ACM Transactions on Modeling and Computer Simulation 2(3). ACM (1992) 195–213
10. Alexander, J., Beck, J. and Chen, W.: Geometric Discrepancy Theory and Uniform Distribution. In: Handbook of Discrete and Computational Geometry. CRC Press (1997) 185–208
11. American-Mathematical-Society: The most irrational number. http://www.ams.org/new-in-math/cover/irrational3.html. 2001
12. Owen, A.: Monte Carlo Variance of Scrambled Net Quadrature. In: Journal of Numerical Analysis 34(5). Society for Industrial and Applied Mathematics (1997) 1884–1910
13. Liao, Y. and Sun, C.: An Educational Genetic Algorithm Learning Tool. In: IEEE Transactions on Education 44(2) 2001 CD-ROM Directory 14