Optimizing Low-Discrepancy Sequences with an Evolutionary Algorithm

François-Michel De Rainville
Laboratoire de vision et systèmes numériques
Département de génie électrique et de
génie informatique, Université Laval
Québec (Québec), Canada G1V 0A6
francois-michel.de-rainville.1@ulaval.ca

Christian Gagné
Laboratoire de vision et systèmes numériques
Département de génie électrique et de
génie informatique, Université Laval
Québec (Québec), Canada G1V 0A6
christian.gagne@gel.ulaval.ca

Olivier Teytaud
Équipe TAO – INRIA Saclay - Île-de-France
LRI, Bât. 490, Université Paris Sud
F-91405 Orsay Cedex, France
olivier.teytaud@inria.fr

Denis Laurendeau
Laboratoire de vision et systèmes numériques
Département de génie électrique et de
génie informatique, Université Laval
Québec (Québec), Canada G1V 0A6
denis.laurendeau@gel.ulaval.ca

ABSTRACT

Many fields rely on some stochastic sampling of a given complex space. Low-discrepancy sequences are methods aiming at producing samples with better space-filling properties than uniformly distributed random numbers, hence allowing a more efficient sampling of that space. State-of-the-art methods like nearly orthogonal Latin hypercubes and scrambled Halton sequences are configured by permutations of internal parameters, where permutations are commonly done randomly. This paper proposes the use of evolutionary algorithms to evolve these permutations, in order to optimize a discrepancy measure. Results show that an evolutionary method is able to generate low-discrepancy sequences of significantly better space-filling properties compared to sequences configured with purely random permutations.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Probabilistic algorithms (including Monte Carlo); I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Performance

Keywords

Quasi-random, Orthogonal Latin Hypercube, Scrambled Halton, Discrepancy, Evolutionary Algorithms, Combinatorial Optimization

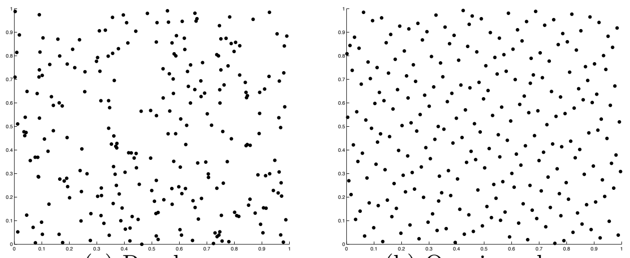
(a) Random (b) Quasi-random

Figure 1: Space filling properties of (a) random numbers can be significantly poorer than low-discrepancy sequences, such as the one generated with a (b) scrambled Halton sequence.

1. INTRODUCTION

Fields such as optimization, Monte Carlo simulations or design of experiments often imply exploring complex high dimensional real-valued spaces. In general, testing all possibilities is computationally intractable and sampling elements of that complex space is an appealing strategy. Pseudo-random number generators are commonly used to sample the space, as they are quite standard and provide numbers following usual probability functions. But such sampling should be done carefully, as the distribution of samples can greatly influence the results obtained. Moreover, the volume of the search space increases exponentially with the number of variables, such that the number of samples required tend to explode with the dimensionality of the problem.

Random point selection can be very disappointing as samples can be distributed in a very non-uniform manner, as shown in Fig. 1(a). It is not possible to (repetitively) obtain something as regular as Fig. 1(b). Quasi-random number generators provide samples with distributions that maximize some space-filling properties while minimizing the correlation between the variables. These approaches have been studied in many areas of computer science, for example in integration [16], optimization [1], and learning [3].

Discrepancy is generally the accepted method to calculate space-filling. It is a measure of proportional distribution of samples in space, where a given region of the entire space must contain a number of samples proportional to its volume. Broadly speaking, discrepancy is, for a rectangle, the absolute ratio of the number of samples contained in the rectangle with the total number of samples minus the ratio of the volume of the rectangle with the entire volume. Given the importance of the notion of discrepancy, quasi-random numbers are also called low-discrepancy sequences.

Two methods are investigated in the current paper: Nearly Orthogonal Latin Hypercubes (NOLH) and Scrambled Halton Sequences (SHS). These methods are useful in different contexts. On one hand, NOLH provide samples that best cover the search space, while minimizing the number of samples used and assuring a high degree of orthogonality between dimensions. This is very useful in situations such as design of experiments, where each sample implies expensive computation. On the other hand, SHS can provide as many samples as necessary. This is appropriate for situations where each sample does not involve heavy processing, or when the number of required samples may be important or unknown in advance. It is possible in many cases to replace standard random number generators with SHS, and then do a possibly more efficient exploration of the complex space sampled. The currently accepted construction methods for those two techniques is based on a randomly shuffled vector of indices. This paper shows that replacing random shuffling by an evolutionary algorithm can lead to significant improvement of the discrepancy and related mesures.

The structure of the paper goes as follows. Sec. 2 presents how NOLH are constructed, while method for building SHS is presented in Sec. 3. Then, discrepancy measure, used to assess space-filling of quasi-random sequences, is presented in Sec. 4, as well as some specific criteria used to assess specific properties of NOLH. Follows in Sec. 5 a presentation of the evolutionary algorithm proposed to configure NOLH and SHS, including the representation, variation operators, and fitness measures used. Experiments with this evolutionary algorithm are reported in Sec. 6, with comparisons made with results obtained using random permutations, showing the efficiency of the proposed technique. The paper finally concludes in Sec. 7 on the possible impact of the proposed method and the future work envisioned.

2. NEARLY ORTHOGONAL LATIN HYPERCUBES

Latin square sampling is defined as a distribution of two variables containing one and only one sample in each of its rows and columns. The Latin hypercube is the generalization of this concept to higher dimensions. It was introduced by McKay *et al.* [15] and has been widely used in computer experiments since then. One weakness of this design is that it does not guarantee that variables will not be correlated, something undesirable when applying regression analysis. Ye [22] proposed a construction method of orthogonal Latin hypercubes that guarantees zero correlation between variables. Cioppa and Lucas [6] extended this algorithm to incorporate more variables while maintaining a certain degree of orthogonality. This last design generates the so called Nearly Orthogonal Latin Hypercubes (NOLH). Cioppa and Lucas' algorithm is used as the base of the work presented in this paper, and is thus further detailed below.

Order (m)	Dimensionality (k)	Runs (n)
4	7	17
5	11	33
6	16	65
7	22	129

Table 1: Dimensionality k and number of runs n as function of the order m of a nearly orthogonal Latin hypercube.

2.1 Constructing NOLH

A NOLH is built from a column vector $\mathbf{e}$, two matrices $\mathbf{M}$ and $\mathbf{S}$ and a set of matrices $\mathcal{A}$. The vector $\mathbf{e}$ is a permutation of the values $[1\ 2\ \cdots\ q]$, where q is the number of positive levels in the NOLH. The dimensionality of a NOLH is obtained by changing an order parameter noted m. For a given order value, some variables can be computed:

$$\begin{aligned} k &= m + \binom{m-1}{2}, && (1)\\ n &= 2^m + 1, && (2)\\ q &= \frac{n-1}{2} = 2^{m-1}. && (3) \end{aligned}$$

Using an NOLH of order m, k variables can be examined, which corresponds to the maximum dimensionality of the search space that can be sampled. The number of runs (levels) n corresponds to the number of samples to be tested for an NOLH of order m. Tab. 1 shows different m associated with the number of variables k it can explore.

First, the set $\mathcal{A}$ is composed of $m-1$ permutation matrices of size $q \times q$, where each matrix is denoted as $\mathbf{A}_i$, with $i = 1, 2, \ldots, m-1$. Matrix $\mathbf{I}$ is a 2×2 identity matrix $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ while matrix $\mathbf{R}$ is a 2×2 matrix $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$.

$$\mathbf{A}_i = \underbrace{\mathbf{I} \otimes \ldots \otimes \mathbf{I}}_{m-1-i} \otimes \underbrace{\mathbf{R} \otimes \ldots \otimes \mathbf{R}}_{i}, \qquad (4)$$

where $\otimes$ is a Kronecker product of matrices.

Then, $\mathbf{M}$, a $q \times k$ matrix, is generated using vector $\mathbf{e}$ and matrices of $\mathcal{A}$. First column $\mathbf{m}_1$ corresponds to the vector $\mathbf{e}$, while, for the next $m-1$ columns, $\mathbf{m}_i$ is obtained by multiplying $\mathbf{A}_i$ with $\mathbf{e}$. Remaining columns $\mathbf{m}_{m+1}, \ldots, \mathbf{m}_k$ correspond to every pairwise multiplication of the matrices $\mathbf{A}_j$ and $\mathbf{A}_l$ with the vector $\mathbf{e}$, with $j = 1, \ldots, m-2$ and $l = j+1, \ldots, m-1$.

Afterwards, $\mathbf{S}$, also a $q \times k$ matrix, is constructed as follows. The values in the first column $\mathbf{s}_1$ are set to 1. The subsequent $m-1$ columns, $\mathbf{s}_2, \ldots, \mathbf{s}_m$, are alternating between -1 and 1, to reflect the estimation of the main effect in a two-level full factorial design. The remaining columns, $\mathbf{s}_{m+1}, \ldots, \mathbf{s}_k$, are the pairwise element-wise multiplication of the $\mathbf{s}_2$ to $\mathbf{s}_m$ columns.

Next, matrix $\mathbf{T}$ is built with the Hadamard product between matrices $\mathbf{M}$ and $\mathbf{S}$. Finally, the sampling matrix is an augmentation of matrix $\mathbf{T}$ with its mirror negative image and a centre point. A sampling matrix for the hypothetical matrix $\mathbf{T} = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$ would be $\left[\begin{smallmatrix} a & c & 0 & -a & -c \\ b & d & 0 & -b & -d \end{smallmatrix}\right]^T$. Interested readers are referred to [6] for more details on NOLH.

2.2 Florian's Method

Florian's method is a routine that is applied to the final sampling matrix in order to decrease its maximum pairwise correlation. Briefly, it exchanges the position of two levels within the same column of the sampling matrix until there cannot be anymore improvement of the correlation in the matrix. Cioppa and Lucas use this procedure only on the sampling matrices that pass a certain screening test. In the current paper, when Florian's method is used, it is applied on every sampling matrix without discrimination. Interested readers are referred to [6, 10] for more details on Florian's method.

3. SCRAMBLED HALTON SEQUENCES

Producing a Scrambled Halton Sequence (SHS) provides an unbounded number of samples for a given dimensionality. Such a low-discrepancy sequence can be very interesting to functionally replace a sequence obtained from usual pseudo-random number generators.

3.1 Van Der Corput Sequences

Consider p to be a prime number. The following sequence generates the n^{th} element $x_{n,p} \in [0,1]$ of the Van Der Corput sequence in basis p:

- Write n in basis p: $n = d_k d_{k-1} \cdots d_1$, that is $n = \sum_{i=1}^{k} d_i p^{i-1}$ with $d_i \in \{0, \ldots, p-1\}$;
- $x_{n,p} = 0.d_1 d_2 \cdots d_k$ in basis p, i.e. $x_{n,p} = \sum_{i=1}^{k} d_i p^{-i}$.

A classical improvement to such sequence, called *scrambling*, defines $x_{n,p}$ as:

$$x_{n,p}^{\pi} = 0.\pi(d_1)\pi(d_2)\cdots\pi(d_k), \tag{5}$$

where π is a permutation of $[0\ 1\ \cdots\ p-1]$ such that $\pi(0) = 0$ in order to ensure that $\forall n > 0,\ x_{n,p} \neq 0$. Reverse scrambling consists in using $\pi(0) = 0$ and for $j > 1$, $\pi(j) = p - j$.

3.2 Halton Sequences

The Halton sequence generalizes the Van Der Corput sequence to dimension d. Consider p_i the i^{th} prime number. Then, $\mathbf{x}_n$, the n^{th} element of a Halton sequence in dimension d, is:

$$\mathbf{x}_n = (x_{n,p_1}, x_{n,p_2}, \ldots, x_{n,p_d}) \in [0,1]^d. \tag{6}$$

The Halton sequence ensures the $O(\log(n)^d/n)$-property for low-discrepancy, as defined later (Eq. 8). However, the sequences can be disappointing, especially for small numbers of points or large dimensionality. In particular, the exponential behaviour in d is problematic – such an exponent does not exist for random independent uniform sequences.

Many solutions have been proposed for dealing with large dimensionality when the distribution to be approximated has strong dependencies between coordinates (see e.g. [12]) or when a reasonable grouping is possible [17]. But for $[0,1]^d$ with d large the only results are (see [18, 21] and references therein):

- A deterministic sequence for which it has been proven that the exponential dependency in d is removed, but the discrepancy decreases as $O(1/(\#P)^{\alpha})$ with $\alpha < \frac{1}{2}$, i.e. the sequence is worse than a random independent sequence.
- An existence proof, without any construction, of a sequence that achieves $O(1/(\#P)^{\alpha})$ with $\alpha > \frac{1}{2}$, i.e. there exists something that (i) is better than random, and (ii) is better than standard low-discrepancy in large dimension, but we are not able to build it.

3.3 Scrambling Halton Sequences

Scrambling is a technique for improving the discrepancy, in particular for a moderate number of examples. With scrambling, $\mathbf{x}_n$ is replaced by:

$$\mathbf{x}_n^{\pi} = \left(x_{n,p_1}^{\pi(1)}, x_{n,p_2}^{\pi(2)}, \ldots, x_{n,p_d}^{\pi(d)}\right) \in [0,1]^d. \tag{7}$$

As many works on the topic, the method proposed here focuses on scrambling values given in Eq. 7.

Various scrambling fitting the framework of Eq. 7 have been proposed [20], each of them providing a new sequence of permutations $(\Pi_n)_{n\in\mathcal{N}}$ (Π_n is a permutation of $[0\ 1\ \cdots\ (p_n-1)]$ with 0 as fixed point). It was then shown that in spite of its simplicity, the simple reverse scrambling (i.e. $\Pi_n = [0\ (p_n-1)\ (p_n-2)\ \cdots\ 1]$) is almost always as efficient, and often better than the other techniques.

4. SAMPLING MATRIX EVALUATION

Cioppa and Lucas [6] present two different properties of a sampling matrix: space-filling and orthogonality. The former is measured by the modified second level (ML_2) discrepancy and the Euclidean maximin (Mm) distance while the latter is measured by the condition number and the maximum pair-wise correlation. The key for a good design is to minimize the condition number, maximum pairwise correlation and ML_2 discrepancy while maximizing the Mm distance.

4.1 Discrepancy

The various measures quantifying the distance between a point set and uniformity are expressed by a discrepancy measure. There are various forms of discrepancy; the most well known is the star-discrepancy [16], defined as follows for a matrix $\mathbf{x}$:

$$D^*(x) = \sup_{u\in[0,1]^k} \Bigg| \#\Big\{ i \in \{1,\ldots,n\}, \\ \forall_{j\in\{1,\ldots,k\}}\ x_{i,j} < u_j \Big\} - \prod_{j=1}^{k} u_j \Bigg|, \tag{8}$$

where $\#E$ is the cardinal of the set E. It is known [5] that the discrepancy of uniform random independent points verifies:

$$\limsup_{n\to\infty} \frac{\sqrt{2n}D^*(x)}{\sqrt{\log(\log(n))}} = 1, \tag{9}$$

whereas low-discrepancy sequences verify:

$$\limsup_{n\to\infty} \frac{nD^*(x)}{\log(n)^k} = 1, \tag{10}$$

(the exponent for the logarithm is $k-1$ in some cases, namely when the number of points is known in advance, with e.g. Hammersley points). The difference between Eq. 9 and Eq. 10 is the main reason for choosing quasi-random points instead of random or pseudo-random points. However, the star-discrepancy has the following drawbacks:

- The supremum involved in Eq. 8 neglects the uniformity in several parts of the domain (only the points close to the frontiers of the u realizing the supremum matter);

- As Fang *et al.* [8] state "one disadvantage of [the star-discrepancy] measure is that it is expensive to compute".

Therefore, Hickernell [12] proposed various other discrepancies, in particular the ML_2 discrepancy. The ML_2 discrepancy is close to the star-discrepancy [9], but replaces the supremum norm by a L_2 norm and averages the result over projections of the design on subsets of coordinates. Importantly, it can be rewritten as follows on a sampling matrix with entries in the range $[0,1]$:

$$ML_2 = \left(\frac{4}{3}\right)^k - \frac{2^{1-k}}{n}\sum_{d=1}^{n}\prod_{i=1}^{k}\left(3 - x_{d,i}^2\right) + \frac{1}{n^2}\sum_{d=1}^{n}\sum_{j=1}^{n}\prod_{i=1}^{k}\left[2 - \max\left(x_{d,i}, x_{j,i}\right)\right], \quad (11)$$

where $x_{i,j}$ is an entry in the sampling matrix. A larger discrepancy means that there are more points in a certain area of the design and leads to a poorer space-filling property.

4.2 Euclidean Maximin Distance

The Euclidean maximin distance (*Mm*) is defined to be the smallest distance between a pair of rows in the sampling matrix. Let $\mathbf{d} = [d_{1,2}\ d_{1,3}\ \cdots\ d_{n-1,n}]$ where $d_{i,j}$ denotes the Euclidean distance between rows i and j, $i < j$:

$$d_{i,j} = \sqrt{(i_1 - j_1)^2 + \ldots + (i_k - j_k)^2}. \quad (12)$$

The *Mm* distance is the smallest $d_{i,j} \in \mathbf{d}$, computed on a sampling matrix with entries in the range $[-1,1]$. A large minimal distance indicates that the distance between the closest point is also large and leads to a better space-filling property.

4.3 Condition Number

Usually the condition number is used in linear algebra to assess whether or not a problem is numerically well-conditioned. Cioppa and Lucas [6] used it to compute the degree of orthogonality of a sampling matrix. A matrix with a condition number equal to 1 is orthogonal while a condition number greater than one indicates some degree of non-orthogonality. The condition number is calculated by the singular value decomposition of $\mathbf{X}^T\mathbf{X}$:

$$\text{cond}_2(\mathbf{X}^T\mathbf{X}) = \frac{\psi_1}{\psi_n}, \quad (13)$$

where $\mathbf{X}$ is the sampling matrix, ψ_1 is the largest singular value and ψ_n is the smallest singular value of $\mathbf{X}^T\mathbf{X}$. As stated by Cioppa and Lucas, a nearly orthogonal sampling matrix would have a condition number no greater than 1.13.

4.4 Maximum Pairwise Correlation

The maximum pairwise correlation is the absolute largest correlation between a pair of columns of the sampling matrix. The correlation between two vectors, $\mathbf{v} = [v_1\ v_2\ \cdots\ v_n]$ and $\mathbf{w} = [w_1\ w_2\ \cdots\ w_n]$, is computed as follows:

$$MPwC = \max_{\forall v, \forall w, v<w}\left|\frac{\sum_{i=1}^{n}\left[(v_i - \overline{v})(w_i - \overline{w})\right]}{\sum_{i=1}^{n}(v_i - \overline{v})^2\sum_{i=1}^{n}(w_i - \overline{w})^2}\right|. \quad (14)$$

According to Cioppa and Lucas [6], the maximum pairwise correlation of a NOLH would be no greater than 0.03.

5. EVOLVING RANDOM PERMUTATIONS

Evolutionary algorithms have already been use to evolve optimized Latin hypercubes [2, 13]. Such Latin hypercubes are closer to McKay's [15] Latin hypercubes than Ye's [22] orthogonal Latin hypercubes as they can contain as many samples as needed. That difference leads to a different encoding. Bates *et al.* [2] and Liefvendahl and Stocki [13] encode directly, in their own manner, the sampling matrix in the genotype. In the present work, we use the construction algorithms presented in Sec. 2.1 and 3 using vector $\mathbf{e}$ as the sampling matrix's genotype for NOLH, and permutation of indices associated to prime numbers in SHS. In our opinion, evolving permutations of indices with an evolutionnary algorithm should allow to move from an initial population of good random low-discrepancy sequences, equivalent to solutions obtained with random permutations, to excellent configurations optimizing the permutations according to discrepancy and related measures presented in Sec. 4.

5.1 Representation

The evolved population is made of individuals represented as fixed-length integer-valued vectors. These vectors are permutation of indices, that is $[a_1\ a_2\ \cdots\ a_n]$, where n is the size of the genotype, and $a_i \in \{1, 2, \ldots, n\}$ an index, with $a_i \neq a_j$, $\forall i \neq j$. That genotype, which is manipulated by the variation operators described in Sec. 5.2. The evolved vectors of permutations are used differently for the configuration of NOLH and SHS.

For NOLH, the permutation of indices is directly used as vector $\mathbf{e}$, introduced in Sec. 2.1. This is the only parameter in Cioppa and Lucas' method that can be modified for generating a NOLH of a given fixed order m. The length q of vector $\mathbf{e}$ depends on the order m of the NOLH, with $q = 2^{m-1}$. Given that, the number of possible permutations for a vector $\mathbf{e}$ of length q increases as $O(2^m!)$, it is computationally intractable to try all possible permutations with $m > 4$. Cioppa and Lucas propose to perform random permutations to explore some of the possible configurations of NOLH. Random permutations are replaced here by an evolutionary algorithm searching for more efficient permutations.

Sec. 3 explains how SHS are constructed from permutations of indices $\Pi_i = [0\ \pi_i(1)\ \cdots\ \pi_i(p_i - 1)]$, where p_i is the i^{th} prime number. In a given permutation of indices Π_i, $\pi_i(0) = 0$ is used as fixed point, while components $\pi_i(j)$, $j > 0$ are the other permuted indices, with $\pi_i(j) \neq \pi_i(k)$, $\forall k \neq j$. Therefore, for constructing a SHS generating samples of dimensionality d, it is necessary to provide $d - 1$ different permutations of sequences Π_i – permutation sequence Π_1 for prime number $p_1 = 2$ is necessarily the identity and need not be evolved. According to this, the evolved genotype is a permutation of indices vector that is transcripted into a phenotype representing the different Π_i, $i = 2, \ldots, d$. That transcription is done by assigning the $p_i - 1$ indices in the genotypes to a sequence Π_i, using

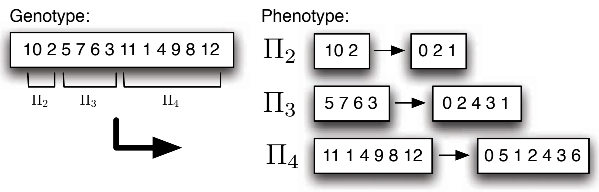

Figure 2: Example of a genotype of indices permutation transcripted into a phenotype of several small indices permutation sequences Π_i, which are used to parameterize scrambled Halton sequences.

```
• Input: Parent individuals, $\mathbf{x}_1 = [x_{1,1} \cdots x_{1,n}]$ and $\mathbf{x}_2 = [x_{2,1} \cdots x_{2,n}]$.
• Output: Offspring individuals, $\mathbf{y}_1 = [y_{1,1} \cdots y_{1,n}]$ and $\mathbf{y}_2 = [y_{2,1} \cdots y_{2,n}]$.

1. Initialize offspring, $\mathbf{y}_1 = \mathbf{y}_2 = \emptyset$, and counters, $j_1 = j_2 = 1$.
2. While $j_1 \neq n$ and $j_2 \neq n$:
   (a) Select index value $z$ from parents:
       • If $j_1 = n \rightarrow z = x_{2,j_2}$ and $j_2 = j_2 + 1$;
       • Else if $j_2 = n \rightarrow z = x_{1,j_1}$ and $j_1 = j_1 + 1$;
       • Else select randomly a parent $k \in \{1, 2\}$, and set $z = x_{k,j_k}$ and $j_k = j_k + 1$.
   (b) Add index value $z$ to an offspring:
       • If $z \in \mathbf{y}_1 \rightarrow \mathbf{y}_2 = [\mathbf{y}_2 \; z]$;
       • Else if $z \in \mathbf{y}_2 \rightarrow \mathbf{y}_1 = [\mathbf{y}_1 \; z]$;
       • Else select randomly offspring $k \in \{1, 2\}$, and set $\mathbf{y}_k = [\mathbf{y}_k \; z]$.
```

Figure 3: Crossover operator used to generate two offspring individuals by mixing two parent individuals.

the relative value of indices in the genotype to infer the new indices of the sequence. Fig. 2 illustrates this process, with a concrete example for $d = 4$. For evolving a configuration of SHS of dimension d, genotypes of size $\sum_{i=2}^{d}(p_i - 1)$ are necessary. It should be noted that many different genotypes of the same size can lead to identical phenotypes. This redundant representation may have interesting effects on the evolutionary search. This is a question of interest that has not been investigated in the current paper, for space and scope reasons.

5.2 Variation Operators

Crossover and mutation have been used as variation operators to generate offspring from parents. Both operators are designed to preserve the fact that genotypes represent permutation of indices, such that every index value should appear only once in each individual. The crossover operator proceeds by randomly mixing index values of two parents to generate two offspring, as presented in Fig. 3. The mutation operator consists in swapping values in a genotype according to a probability value, as presented in Fig. 4.

```
• Input: Original individual, $\mathbf{x} = [x_1 \cdots x_n]$ and swapping probability, $\rho$.
• Output: Mutated individual, $\mathbf{y} = [y_1 \cdots y_n]$.

1. Initialize mutated individual as original individual, $\mathbf{y} = \mathbf{x}$.
2. For $i = 1, \ldots, n$:
   (a) If $z < \rho$, with $z = \mathcal{U}(0, 1)$ a random value draw uniformly in $[0, 1]$, then:
       • Select randomly an index $k \in \{1, \ldots, (i-1), (i+1), \ldots, n\}$ to swap;
       • Swap values $y_i$ and $y_k$ of mutated individual.
```

Figure 4: Mutation operator for swapping index values of individual.

5.3 Fitness Measures

The fitness $\mathrm{f}_{\mathrm{NOLH}}$ of an individual $\mathbf{x}$ used for NOLH is composed of four measures:

$$\mathrm{f}_{\mathrm{NOLH}}(\mathbf{x}) = \{ML_2(\mathbf{x}), Mm(\mathbf{x}), \mathrm{f}_{Cond}(\mathbf{x}), \mathrm{f}_{Cor}(\mathbf{x})\}, \quad (15)$$

$$\mathrm{f}_{Cond}(\mathbf{x}) = \min(1, 1.13/\operatorname{cond}_2(\mathbf{x})), \quad (16)$$

$$\mathrm{f}_{Cor}(\mathbf{x}) = \min(1, 0.03/MPwC(\mathbf{x})), \quad (17)$$

where $ML_2(\mathbf{x})$ is the ML_2 discrepancy (Sec. 4.1) of the samples generated by constructing a NOLH with individual $\mathbf{x}$, $Mm(\mathbf{x})$ is the Euclidean Mm distance (Sec. 4.2) of the corresponding NOLH, $\operatorname{cond}_2(\mathbf{x})$ is the condition number (Sec. 4.3) associated with that NOLH, and $MPwC(\mathbf{x})$ is the corresponding maximum pairwise correlation (Sec. 4.4). The last two values are bounded by the prescribed values of near orthogonality (i.e. $\operatorname{cond}_2(\mathbf{x}) \leq 1.13$ and $MPwC(\mathbf{x}) \leq 0.03$). A relative selection algorithm is used (i.e. k-participants tournament selection), such that comparison of two individuals is made by a normalized sum of the four fitness measures:

$$\mathrm{g}(\mathbf{x}_1, \mathbf{x}_2) = \frac{ML_2(\mathbf{x}_2) - ML_2(\mathbf{x}_1)}{ML_2(\mathbf{x}_1) + ML_2(\mathbf{x}_2)} + \frac{Mm(\mathbf{x}_1) - Mm(\mathbf{x}_2)}{Mm(\mathbf{x}_1) + Mm(\mathbf{x}_2)} + \frac{\mathrm{f}_{Cond}(\mathbf{x}_1) - \mathrm{f}_{Cond}(\mathbf{x}_2)}{\mathrm{f}_{Cond}(\mathbf{x}_1) + \mathrm{f}_{Cond}(\mathbf{x}_2)} + \frac{\mathrm{f}_{Cor}(\mathbf{x}_1) - \mathrm{f}_{Cor}(\mathbf{x}_2)}{\mathrm{f}_{Cor}(\mathbf{x}_1) + \mathrm{f}_{Cor}(\mathbf{x}_2)}, \quad (18)$$

with $\mathbf{x}_1$ being selected when $\mathrm{g}(\mathbf{x}_1, \mathbf{x}_2) \geq 0$, otherwise $\mathbf{x}_2$ is selected. This complex fitness comparison approach is adopted in order to allow the comparisons of our results with those obtained by Cioppa and Lucas [6].

For SHS, the fitness consists of a direct minimization of the ML_2 discrepancy (Sec. 4.1). This is the quality criterion used by Vandewoestyne and Cools [20] to assess approaches to generate low-discrepancy sequences.

6. EXPERIMENTS

This section presents the experiments conducted on the optimization of the NOLH with an Evolutionary Algorithm (NOLH-EA) in Sec. 6.1, as well as the optimization of SHS with an Evolutionary Algorithm (SHS-EA) in Sec. 6.2. In both cases, most of the computational time was taken by accessing candidate solutions, with negligible overhead added by other operations of the evolutionary algorithm. There-

Parameter	NOLH-EA	SHS-EA
Number of generations	150	1500
Population size	10 000	500
Selection type	Tournaments	Tournaments
Participants to tournaments	5	10
Crossover probability	0.5	0.5
Mutation probability	0.3	0.3
Mutation swapping probability (ρ)	0.2	0.02

Table 2: Parameters of the evolutionary algorithm used for the evolution of permutations for NOLH-EA(-FLO) and for SHS-EA.

Rank	ML_2	Mm	f_{Cond}	f_{Cor}
1	0.688531	1.83286	1	1
2	0.683901	1.81573	1	1
3	0.691428	1.83286	1	1
4	0.692991	1.83286	1	1
5	0.683289	1.80701	1	1
6	0.693695	1.82324	1	1
7	0.706094	1.83286	1	1
8	0.701136	1.81681	1	1
9	0.699427	1.80710	1	1
10	0.708548	1.82645	1	1
Mean	0.6948	1.8228	–	–
Std. dev.	0.0087	0.0105	–	–

Table 3: Ten best designs found over 50 experiments using an evolutionary algorithm, without using Florian's method (NOLH-EA).

Rank	ML_2	Mm	f_{Cond}	f_{Cor}
1*	0.731822	1.75780	1	1
1	0.736484	1.76777	1	1
2	0.722773	1.72187	1	1
3	0.687863	1.63339	1	1
4	0.732031	1.73543	1	1
5	0.737450	1.71733	1	1
6	0.718618	1.78645	1	0.935
7	0.724820	1.68402	1	1
8	0.710764	1.65123	1	1
9	0.739440	1.70706	1	1
10	0.744426	1.70706	1	1
NA**	0.733508	1.78754	0.918	0.408

Table 4: Ten best designs found after 12.5 million random permutations (NOLH-RND) : * denotes Cioppa and Lucas' best design using Florian's method, ** denotes Cioppa and Lucas' best design without using Florian's method.

Rank	ML_2	Mm	f_{Cond}	f_{Cor}
1	0.660880	1.93548	1	1
2	0.650608	1.89572	1	1
3	0.670614	1.94856	1	1
4	0.656370	1.89263	1	1
5	0.656127	1.89159	1	1
6	0.650319	1.87916	1	0.99733
7	0.662342	1.90702	1	1
8	0.660529	1.90189	0.99912	1
9	0.651672	1.87292	1	1
10	0.666234	1.90907	0.99975	1
Mean	0.6586	1.9034	–	–
Std. dev.	0.0068	0.0234	–	–

Table 5: Ten best designs found over 50 different experiments with an evolutionary algorithm using Florian's method (NOLH-EA-FLO).

fore, the evolutionary techniques can be compared to the random permutation ones according to the number of solutions accessed.

6.1 Results with NOLH

Two sets of 50 experiments have been performed with the evolution of NOLH of order $m = 5$, dimensionality $d = k = 11$ and $n = 33$ samples. The first set constructs a NOLH as explained in Sec. 2.1, the second set adds Florian's method (Sec. 2.2) to this construction procedure. The results for the first set of experiments are compared to the results obtained with 12.5 million random permutations. The results for the second set of experiments are compared to Cioppa and Lucas' best design using Florian's method. Evolutions have been conducted using parameters presented in Tab. 2. All experiments have been implemented using the Open BEAGLE C++ framework for evolutionary computation [11].

The results of the first set of experiments, constructing the NOLH with an evolutionay algorithm and without using Florian's method (NOLH-EA) are shown in Tab. 3. Tab. 4 presents the results of the random search and Cioppa and Lucas' best design. In the tables, f_{Cond} and f_{Cor} refer to the condition number and the maximum pairwise correlation respectively, as described in Sec. 5.3. On average, NOLH-EA designs improve discrepancy by 5.7% and maximin distance by 3.1% compared to the best NOLH-RND design. In addition, the computational effort needed to reach these solutions is lower, with an average of 1 million solutions tested for each run of the evolutionary algorithm, compared to the 12.5 million solutions tested by the random search.

Above experiments have been performed without using Florian's method. Including this method in the evolutionary algorithm provides better results as shown in Tab. 5, which presents the ten best designs of the second set of experiments. On average, these designs have a discrepancy and maximin distance that are respectively 10.0% and 8.3% better than Cioppa and Lucas' best design using Florian's method (Fig. 5). The best sampling matrix built with a evolutionary algorithm and Florian's method (NOLH-EA-FLO) can be seen in Fig. 6. It can be observed that points obtained with NOLH-EA-FLO are distributed more uniformly than with Cioppa and Lucas' method for which poor space-filling visual patterns, like the "X" shaped plot between variables A-I, C-D, F-G, and J-K, is visible.

6.2 Results with SHS

For SHS of dimensionality $d = 11$ and $n = 200$ samples, 20 independent experiments are made using an evolutionary algorithm (SHS-EA) and the configuration shown in Tab. 2. The ten best results obtained from these runs are compared to the results of 12.5 million randomly generated scrambled Halton sequences (SHS-RND). Tab. 5(a) shows the ten best

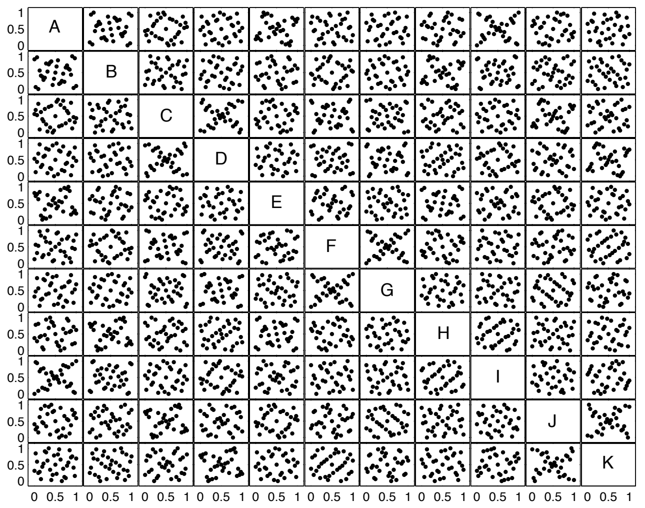

Figure 5: Pairwise dimension representation for the best design found by Cioppa and Lucas using Florian's method.

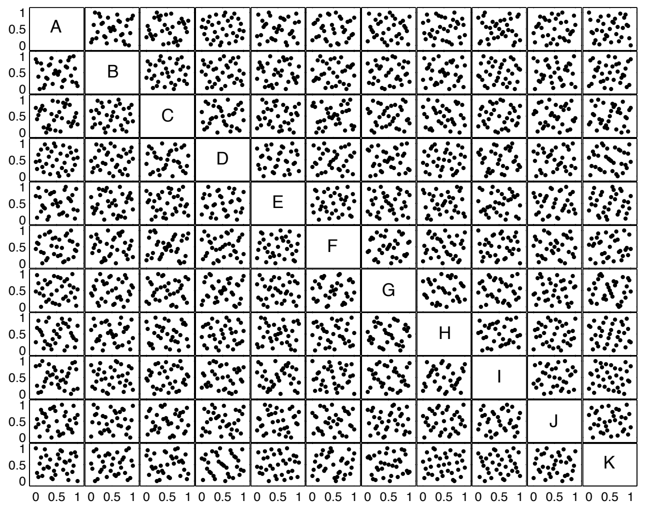

Figure 6: Pairwise dimension representation for the best design found with an evolutionary algorithm using Florian's method (NOLH-EA-FLO).

randomly generated SHS, as well as the SHS generated by reverse scrambling [20]. Tab. 5(b) presents a list of the ten best results obtained from the 20 independent experiments made with an evolutionary algorithm.

It is observed that SHS-EA outperforms SHS-RND by 12.9%, on average. The best sequences obtained with SHS-RND and SHS-EA are presented in Fig. 7 and 8, respectively. The overall uniformity of SHS-EA is slightly superior to SHS-RND, the plot of the latter showing non uniform patterns between variables B-C, and F-G, while such patterns are almost absent of SHS-EA. Once again, the computational effort to find these better solutions is lower, with an average of 500 000 sequences explored by the evolutionary algorithm compared to the 12.5 million tested for the random case.

7. CONCLUSION

This paper has investigated the possibility of generating low-discrepancy sequences using an evolutionary algorithm. In both studied cases, NOLH and SHS, the proposed evolutionary algorithm significantly outperforms the original random technique. Also in both cases, lower computational resources are used, showing that the space of all possible sampling matrices can be explored efficiently. Generating sequences of samples with very low discrepancy can be highly beneficial to many scientific fields such as finance [19], volume and surface calculation [7], and even evolutionary algorithms [1]. A practical motivation of our work is to exploit low-discrepancy sequences in the context of computer assistance to increase human comprehension of military complex situations [14]. In fact, any domain relying on well-distributed sampling of a given space can eventually benefit from the techniques proposed in the current paper to increase their efficiency.

The current work can be extended in many ways. Firstly, new crossover and mutation operators could be developed specifically for an efficient evolution of low-discrepancy sequences. Secondly, other discrepancy measures can be tested. For example, non-extreme or centred versions of the discrepancy can be considered in order to avoid poor uniformity and ensure symmetry (see [12] and [16, page 14]). In many cases, uniformity measures are somewhat disappointing [4] and could be replaced by performance measures on a set of benchmarks. Thirdly, as there are several measures of uniformity, multiobjective optimization could be used. And finally, instead of optimizing the discrepancy for a fixed number of points in a fixed dimension, some "anytime" properties could be considered: generating permutations which ensure, as far as possible, good properties independently of the number of points and dimensions used.

Acknowledgments

The authors acknowledge the financial support of Defence Research and Development Canada.

8. REFERENCES

[1] A. Auger, M. Jebalia, and O. Teytaud. XSE: quasi-random mutations for evolution strategies. In *Proceedings of Evolutionary Algorithms*, page 12, 2005.

[2] S.J. Bates, J. Sienz, and V.V. Toropov. Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm. In *5th ASMO-UK/ISSMO Conference on Engineering Design Optimization*, 2004.

[3] C. Cervellera and M. Muselli. Deterministic design for neural network learning: an approach based on discrepancy. *IEEE Transactions on Neural Networks*, 15(3):533–544, 2004.

[4] C. Chlier. Error trends in quasi-Monte Carlo integration. *Comp. Phys. Comm.*, 193:93–105, 2004.

[5] K.-L. Chung. An estimate concerning the Kolmogoroff limit distribution. *Transactions of the American Mathematical Society*, 67:36–50, 1949.

[6] T. M. Cioppa and T. W. Lucas. Efficient nearly orthogonal and space-filling latin hypercubes. *Technometrics*, 49(1):45–55, February 2007.

(a) SHS-RND

Rank	ML_2
1	0.0769026
2	0.0770423
3	0.0772918
4	0.0774606
5	0.0775166
6	0.0775733
7	0.0775951
8	0.0776178
9	0.0776438
10	0.0776462
NA*	0.1770522

(b) SHS-GA

Rank	ML_2
1	0.0661650
2	0.0665556
3	0.0666159
4	0.0667767
5	0.0670300
6	0.0672203
7	0.0672460
8	0.0673653
9	0.0675620
10	0.0676506
Mean	0.0670
Std. dev.	4.7971×10^{-4}

Table 6: Experiments for both techniques: (a) Random permutations, where * is the reverse scrambled Halton sequence, and (b) scrambled Halton sequences found with an evolutionary algorithm.

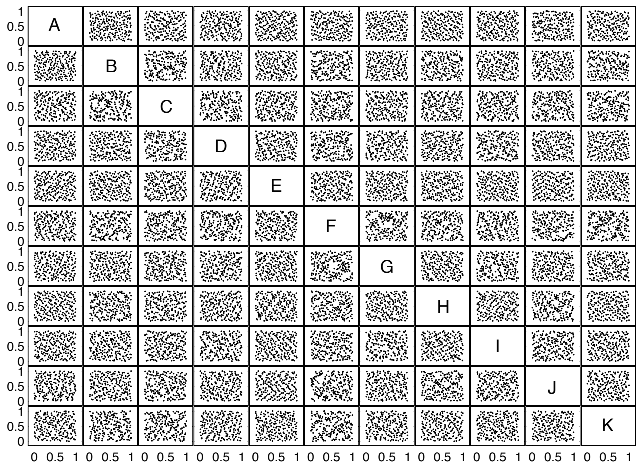

Figure 7: Pairwise dimension representation for the best scrambled Halton sequence found with random permutations (SHS-RND).

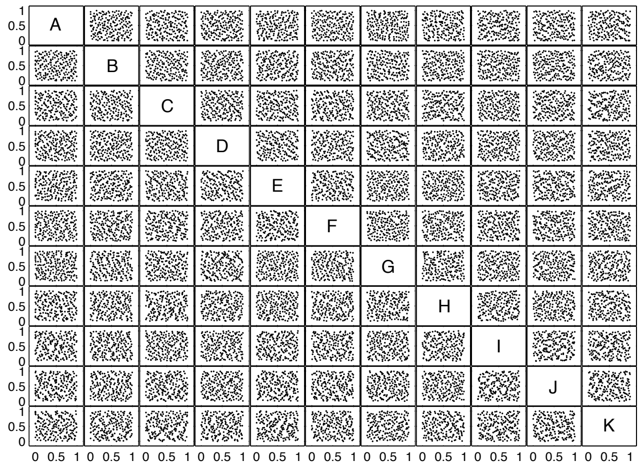

Figure 8: Pairwise dimension representation for the best scrambled Halton sequence found with an evolutionary algorithm (SHS-EA).

[7] T. J. G. Davies, R. R. Martin, and A. Bowyer. Computing volume properties using low-discrepancy sequences. In *Geometric Modelling*, pages 55–72, 1999.

[8] K. T. Fang, D. K. J. Lin, P. Winker, and Y. Zhang. Uniform design: Theory and application. *Technometrics*, 42(3):237–248, August 2000.

[9] K. T. Fang and Y. Wang. *Number-theoretic Methods in Statistics*. Chapman and Hall, 1994.

[10] A. Florian. An efficient sampling scheme : updated Latin hypercube sampling. *Probabilistic engineering mechanics*, 7(2):123–130, 1992.

[11] C. Gagné and M. Parizeau. Genericity in evolutionary computation software tools: Principles and case study. *International Journal on Artificial Intelligence Tools*, 15(2):173–194, Apr. 2006.

[12] F. J. Hickernell. A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, 67(221):299–322, 1998.

[13] M. Liefvendahl and R. Stocki. A study on algorithms for optimization of Latin hypercubes. *Journal of Statistical Planning and Inference*, 136(9):3231–3247, 2006.

[14] M. Lizotte, D. Poussart, F. Bernier, M. Mokhtari, E . Boivin, and M. DuCharme. IMAGE: Simulation for understanding complex situations and increasing future force agility. In *Proceedings of the 26th Army Science Conference*, page 7, 2008.

[15] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, May 1979.

[16] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, 1992.

[17] A. B. Owen. Quasi-Monte Carlo sampling. In H. W. Jensen, editor, *Monte Carlo Ray Tracing: Siggraph 2003 Course 44*, pages 69–88. SIGGRAPH, 2003.

[18] I. H. Sloan and H. Woźniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *Journal of Complexity*, 14(1):1–33, 1998.

[19] K. S. Tan and P. P. Boyle. Applications of randomized low discrepancy sequences to the valuation of complex securities. *Journal of Economic Dynamics and Control*, 24:1747–1782, October 2000.

[20] B. Vandewoestyne and R. Cools. Good permutations for deterministic scrambled Halton sequences in terms of l_2-discrepancy. *Computational and Applied Mathematics*, 189(1,2):341:361, 2006.

[21] G. Wasilkowski and H. Wozniakowski. The exponent of discrepancy is at most 1.4778. *Math. Comp*, 66:1125–1132, 1997.

[22] K. Q. Ye. Orthogonal column Latin hypercubes and their application in computer experiments. *Journal Association Statistical Analysis, Theory and Methods*, 93:1430–1439, 1998.