Extended Deterministic Local Search Algorithm for Maximin Latin Hypercube Designs

Tobias Ebert, Torsten Fischer, Julian Belz, Tim Oliver Heinz, Geritt Kampmann, Oliver Nelles
Department of Mechanical Engineering
University of Siegen
D-57068 Siegen, Germany
Email: http://www.mb.uni-siegen.de/mrt/mitarbeiter/

***Abstract*—This paper introduces the Extended Deterministic Local Search (EDLS) algorithm for Latin Hypercube (LH) designs. The main goal of the algorithm is to improve an existing algorithm towards a better uniformity of the data distribution, while maintaining a good computational performance. Uniform distributed data sets are well suited for system identification if no model structure is known beforehand. After presenting background information about LH designs and how to assess their quality (choice of loss function), the EDLS algorithm is explained and compared to two other algorithms for LH designs.**

I. Introduction

The foundation for creating good models from (measured) data is a good distribution of the data points in the input space, the so-called experimental design. In the field of Design of Experiments (DoE) many methods to find a well-suited design for function approximation have been proposed. This generally means to find a design that, despite of measurement noise, allows for low bias and minimizes the variance in the estimated parameters of the model, while requiring as few data points as possible to minimize experimental costs.

If the model structure corresponds to a linear regression problem, several popular optimality criteria exist (e.g. D-, A-,V-optimal designs) for which an optimal design can be calculated efficiently. For instance, an A-optimal design minimizes the sum of the variances of the regression coefficients [1].

If it is not known what kind of model structure will fit the identification problem best or the model type is unknown beforehand for other reasons, different criteria may be of higher importance. For example an uniform distribution of data points in the entire input space is a reasonable demand in this case. Also a non-collapsing design is desirable, if the importance of individual inputs is unknown and some might later be removed from the model. Non-collapsing means, that if all data points are projected onto one axis, all values along that axis will only occur once. While providing uniformity, simply placing the points on an evenly spaced grid is usually not feasible in situations with a high number of input dimensions. Furthermore, the grid points coincide on all axis-projections. In these situations the use of Sobol sequences [2], [3] is preferable. A strategy that places points on a fine grid but avoids the curse of dimensionality is the Latin Hypercube (LH) design. LH designs have the property of being non-collapsing, but do not inherently provide a uniform data distribution.

To gain a (at least approximately) uniform distribution, the LH design has to be optimized using a loss function suitable to represent the uniformity. This is an intractable complex combinatorial problem, if an exhaustive method is used which simply tries to calculate the loss function for each possible LH configuration. For higher dimensions or a higher number of design points, this is not a feasible approach.

To solve this problem, some researchers use global optimization methods like particle swarm optimization [4], simulated annealing [5] or a special form of evolutionary algorithm [6] for maximizing a distance measure between points. For these methods a continuous loss function is advantageous for optimization. Therefore [5] introduced the so called ϕ_p criterion, which approximates the maximin distance criterion [7]. Local search methods, like in [8], do not need a continuous loss function. They can directly use the maximin distance criterion, although it is a discontinuous loss function. Another approach is a geometrically ordered LH design [9], [10]. An overview of LH design methods can be found in [11].

The algorithm presented in this paper is a local optimization approach based on the Deterministic Local Search (DLS) algorithm presented in [8], where it was shown that in many cases the algorithm is superior to a simulated annealing approach. Here the DLS algorithm is extended by a second phase, to improve the uniformity of the data distribution, while simultaneously improving the performance, keeping the computational demand moderate and retaining the deterministic nature.

Since it is not a simple task to define a suitable quality criterion for LH design optimization, the following section discusses some aspects of the loss functions usually used. In Sec. III LH designs and their properties are briefly introduced, before in Sec. IV the Deterministic Local Search (DLS) algorithm and the proposed extended version (EDLS) are explained. In Sec. V the EDLS algorithm is tested against two other algorithms for LH designs. Besides the DLS we consider the geometrically approach called translational propagation algorithm (TPLHD) developed in [9], which is supposed to approximate the optimal latin hypercube design reasonably well for up to 6 variables. Finally, a conclusion is given.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.63

II. Loss Function

To evaluate the quality of a data distribution, a loss function has to be defined. Here quality refers to the uniformity of the data distribution, since an even coverage of the input space is the best choice, if no prior knowledge about the process of interest is available. All of the presented loss functions are based on distances in the design space. Basically, they depend on one critical pair of points, which delivers the critical distance d_{crit}. The definition of d_{crit} depends on the loss function at hand as will be described in the following. Usually euclidean distance measures are used, but the functions are not restricted to these.

The general idea is to rearrange the data points such that the loss function and thereby the data distribution improves until a so-called optimal design is achieved. Depending on the loss function either maximin (Mm) or minimax (mM) designs arise, that are introduced in [7]. Both loss functions are based on nearest neighbor distances and typically possess many local optima.

Minimax (mM) Loss Function

In a set of $N_{\mathbb{P}}$ potential design points $\mathbb{P}$, a chosen design with $N_{\mathbb{X}}$ points $\mathbb{X}$ is a subset of $\mathbb{P}$. Typically, $N_{\mathbb{X}} \ll N_{\mathbb{P}}$.

When the maximum distance of each $p \in \mathbb{P}$ to the nearest $x \in \mathbb{X}$ is minimized, $\mathbb{X}$ is called a minimax design of $\mathbb{P}$. In other words: In an mM design, the farthest distance of $p \in \mathbb{P}$ to its nearest point $x \in \mathbb{X}$ has a minimal distance. In [7] a mM design is described as follows:

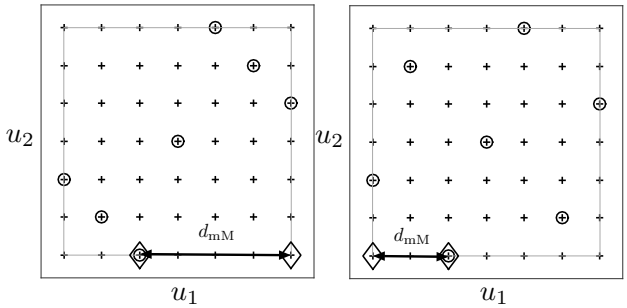
$$\min_{\mathbb{X}} \left(\max_{p \in \mathbb{P}} \left(\min_{x \in \mathbb{X}} (d(p,x)) \right) \right) = d^* \tag{1}$$

Where $d(p,x)$ is the distance between one potential design point p and a chosen point x. It is obvious, that a part of (1) can be used to rate a chosen design $\mathbb{X}_i$:

$$\max_{p \in \mathbb{P}} \left(\min_{x \in \mathbb{X}_i} (d(p,x)) \right) = d_{\mathrm{mM},i} \tag{2}$$

The distance $d_{\mathrm{mM},i}$ rates the design $\mathbb{X}_i$, where smaller values are preferred. The calculation of $d_{\mathrm{mM},i}$ needs $(N_{\mathbb{P}} - N_{\mathbb{X}}) \cdot N_{\mathbb{X}}$ distance evaluations, which makes it not feasible for big $N_{\mathbb{X}}$.

Figure 1 illustrates two different LH designs (see Sec.III). Here the values of $\mathbb{P}$ are represented by crosses and $\mathbb{X}$ are circles. In Fig. 1a one can observe a poor data distribution in two corners. The distances d_{mM} can be interpreted as the distance to the largest "hole" which is penalized in the mM loss function. Figure 1b highlights a mM design. The distance to the biggest "hole" is considerably smaller than in Fig. 1a. Another motivation of the mM design is based on the assumption, that each x delivers an amount of information, which decreases with increasing distance. So in an mM design, the minimal information of each $p \in \mathbb{P}$ is maximized.



(a) Bad LH design (b) mM LH design

Fig. 1. One of the critical distances in terms of a minimax loss function for a bad (a) and optimal (b) design

Maximin (Mm) Loss Function

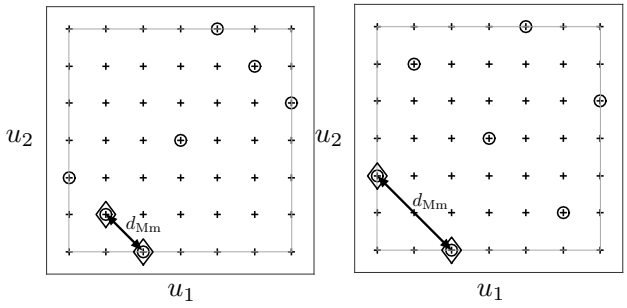
In an Mm design, the distance of the two nearest neighbors of a chosen design is maximized, so that it satisfies the following equation:

$$\max_{\mathbb{X}} \left(\min_{x_i, x_j \in \mathbb{X}, i \neq j} (d(x_i, x_j)) \right) = d^o \tag{3}$$

The distance of the two nearest neighbors in $\mathbb{X}$ can be used to measure the quality of the design.

$$\min_{x_i, x_j \in \mathbb{X}_i, i \neq j} (d(x_i, x_j)) = d_{\mathrm{Mm},i} \tag{4}$$

The calculation of the Mm loss function needs at least $(N_{\mathbb{X}}-1) \cdot N_{\mathbb{X}}/2$ distance evaluations. For $N_{\mathbb{X}} \ll N_{\mathbb{P}}$ the computational effort is significantly smaller than for mM.



(a) Bad LH design (b) mM LH design

Fig. 2. One of the critical distances in terms of a maximin loss function for a bad (a) and optimal (b) design

Figure 2a shows high data concentration in two corners. This density can be represented by the distance d_{Mm}. In the Mm loss function these data clusters are penalized. The goal of an Mm design can be interpreted as follows: Each x delivers an amount of information, which decreases with increasing distance. For an Mm design each point covers a maximal hyper-volume (see d_{Mm} in Fig. 2b) with its information.

Even if the mM design in Fig. 1b and the Mm design in Fig. 2b are equal, this is not the case for more complex designs. As discussed in [7], Mm designs are typically denser around the boundaries.

In case of Mm LH designs the squared critical distance is often used to assess the quality of the data distribution.

$$D^2 = d_{\text{Mm}}^2 \in \mathbb{N} \tag{5}$$

This value can easily be interpreted, if the LH levels incorporate only integer values from 1 up to N. For n input dimensions the minimum diagonal squared distance is known to be n for two points next to each other.

For an LH design $\mathbb{P}$ the loss function of mM and Mm are discontinuous. Both depend on a critical pair of points, which delivers the critical distance d_{crit} that can either be $d_{Mm,i}$ or $d_{mM,i}$. To overcome this discontinuity for the Mm loss function, an approximation is proposed in [5].

Maximin Approximation

It is obvious, that (4) is not continuously differentiable, because changing the critical points, which define $d_{\text{Mm},i}$, would lead to change in the critical point pair. Such a function is hard to optimize. In [5] the ϕ_p value was introduced:

$$\phi_p = \sqrt[p]{\sum_{i=1}^{N}\sum_{j=i+1}^{N}(d(i,j))^{-p}} \tag{6}$$

In contrast to (4), ϕ_p is a continuous function for finite values of p. For $p \to \infty$, the maximin design criterion is reestablished. Note that the higher p is, the more the smallest distances will be emphasized. Besides the property of being a continuous function, the ϕ_p value represents the data distribution better than just the critical distance. All distances contribute to that value. According to [5], [7], [8], typical values are $p = 20\ldots50$. One main weakness of mM and Mm optimization is the sole focus on the worst point. A perfect distribution can be destroyed by a single bad point. This example illustrates the weakness of these quality criteria. Of course their *optimization* drives the points to a "good" overall distribution. However, these criteria are not suited in order to evaluate the quality of any overall point distribution. ϕ_p can weaken but not overcome these shortcomings.

III. Latin Hypercube Designs

Basically there are two desirable properties for the design of experiments, if no prior knowledge about the function or process of interest is available. The design should cover the whole input space uniformly and it should be non-collapsing [12] as explained in the introduction. This property is especially appealing if the relevance of the input variables is unknown a-priori. In case some inputs are identified to be irrelevant, the values of each remaining relevant input are still uniformly spread across each single axis without repetitions.

Latin hypercube (LH) designs were originally proposed for computer experiments by [13] and [14] and automatically fulfill the requirement of being non-collapsing. They are computationally cheap to construct, cover the design region without replications and pursue a geometrical approach to be build [15]. The number of samples N has to be determined a-priori and can be chosen arbitrarily. Once N is defined, each axis of the input space is partitioned into N levels. For n inputs an n-dimensional grid with N^n grid points emerges. Each level of each input is only occupied once, such that a non-collapsing design is guaranteed. Different LH designs for $n = 2$ and $N = 7$ are shown in Figures 1 and 2.

The property of being an LH design does not automatically guarantee a space filling or uniform data distribution. Therefore an optimization of the LH design has to be performed. In order to achieve good space filling properties, one of the loss functions presented in Sec. II can be utilized for the optimization. The optimization of LH designs with respect to a defined loss function leads to so called optimal LH designs [15]. In contrast to the computationally cheap construction of LH designs in general, the computational cost for optimal LH designs is enormous. Thus, the algorithm for finding optimal LH designs should be fast. The optimization procedure proposed in this paper is based on element exchanges in the design matrix and pursues a greedy strategy. It is explained in detail in Sec. IV.

IV. Algorithm

The basis for our study is the local search with coordinate exchange described in [8]. The idea of a coordinate exchange algorithm is straightforward: A set $\mathbb{X}$ of N samples with n variables or dimensions is given. The aim of the DLS algorithm is to maximize the critical distance according to the *maximin* (Mm) criterion in ((3)), see Sec. II. The optimization criterion is improved by swapping the coordinates in one dimension between a pair of samples. Pseudo code for a deterministic local search (DLS) is shown in Algorithm 1. The presented pseudo code is more detailed than necessary, to allow for an easy extension to the code for an improved algorithm, see next subsection. The DLS algorithm features several desirable properties:

- It preserves an LH design, if the initial set of samples is a LH.
- It is able to optimize designs with multiple samples per level, i.e. the algorithm is also able to optimize any collapsing design.
- As long as the range of variables is normalized in a reasonable way, the algorithm operates on mixed discrete and continuous variables.
- It is deterministic, its results are reproducible.
- It ensures improvement in every iteration and thus is guaranteed to converge.

For an example of the algorithm see Fig. 3: a) Initialization: The first critical point pair is 1-2 (the points are numbered according to their columns $1\ldots5$). b) Illegal swap 1 with 3 since the partner point distance to its new nearest neighbor is not bigger than the critical distance: No improvement. c) Legal swap 1 with 4: A distance improvement from 1-1 to 1-2 units. Now the critical point pair is 2-3. Critical distance still is 1-1.

Swapping the first neighbor point 2: d) Illegal swap 2 with 1: The partner point does not improve the critical distance. e) Illegal swap 2 with 4: Swap does not improve critical

Input: A normalized set of points $\mathbb{X}$
Output: Set of points $\mathbb{X}$ with maximized distance of each element to its nearest neighbors
Step 1: Calculate the nearest neighbors $\vec{x}_{i,\mathrm{NN}}$ and its distance $d_{i,\mathrm{NN}}$ to every point $\vec{x}_i \in \mathbb{X}$. Combine all nearest neighbor distances $d_{i,\mathrm{NN}}$ in a vector $\vec{d}_{\mathrm{NN}}$ and calculate the critical distance $d_{\mathrm{crit}} = \min(\vec{d}_{\mathrm{NN}})$;
Step 2:The set of points with $d_{i,\mathrm{NN}} = d_{\mathrm{crit}}$ form the worst set of points $\mathbb{W} \subset \mathbb{X}$;
Step 3: **while** $\mathbb{W} \neq \emptyset$ **do**
  Choose first element of $\mathbb{W}$
  $\vec{w} := \vec{x}_j \mid \vec{x}_j \in \mathbb{W}$;
  Form a set of possible coordinate swap partner points $\mathbb{S} := \mathbb{X} \setminus \{\vec{x}_j, \vec{x}_{j,\mathrm{NN}}\}$;
  while $\mathbb{S} \neq \emptyset$ **do**
    Choose first element of $\mathbb{S}$
    $\vec{s} := \vec{x}_k \mid \vec{x}_k \in \mathbb{S}$;
    Form a set of variables, where coordinates shall be swapped $\mathbb{V} := \{1, \ldots, \mathrm{n}\}$;
    while $\mathbb{V} \neq \emptyset$ **do**
      swap coodinates of $\vec{w}$ and $\vec{s}$ in dimension $v := v_m \mid v_m \in \mathbb{V}$
      resulting in $\widetilde{\vec{w}}$ and $\widetilde{\vec{s}}$;
      Calculate the distances $d_{w,\mathrm{NN}}$ and $d_{s,\mathrm{NN}}$ to their nearest neighbors
      $\widetilde{\vec{w}}_{\mathrm{NN}} \in \mathbb{X}$ and $\widetilde{\vec{s}}_{\mathrm{NN}} \in \mathbb{X}$;
      if $d_{w,NN} > d_{crit}$ *and* $d_{s,NN} > d_{crit}$ **then**
        Update $\mathbb{X}$ by replacing $\vec{x}_k := \widetilde{\vec{s}}$ and $\vec{x}_j := \widetilde{\vec{w}}$;
        goto Step 1 for next iteration.
      else
        Remove v_m from $\mathbb{V}$.
      end
    end
    Remove $\vec{x}_k$ from $\mathbb{S}$.
  end
  Remove $\vec{x}_j$ from $\mathbb{W}$.
end

Algorithm 1: Deterministic Local Search (DLS)

distance. f) Illegal swap 2 with 5: Point and partner point do not improve critical distance. Result: First neighbor point cannot be swapped with any partner in dimension 1.

Swapping the second neighbor point 3: g) Illegal swap 3 with 1: Point does not improve critical distance. h) Illegal swap 3 with 4: Partner point does not improve critical distance. i) Legal swap 3 with 5: Point and partner point improve the critical distance, a success! Critical distance increases from 1-1 to 1-2. Next critical point pair is again 1-2 but no legal swapping partner can be found for point 1 or 2. Next critical point pair is 4-5 but no legal swapping partner can be found for point 4 or 5. Algorithm terminates.

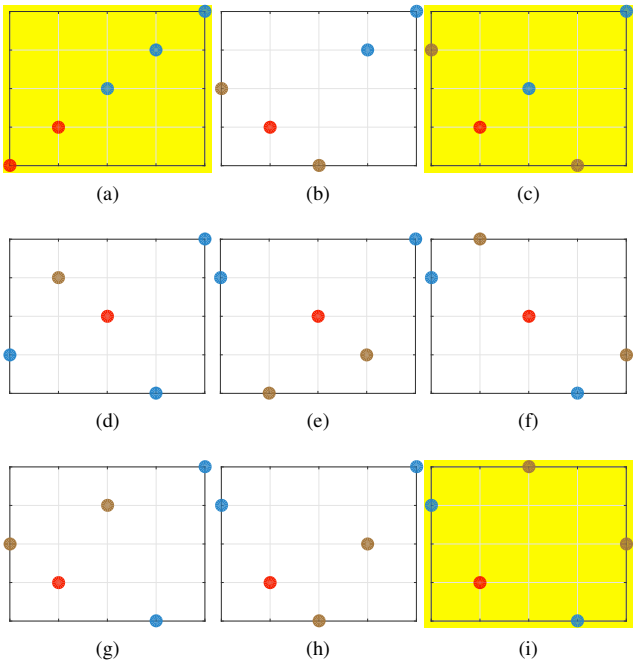


Fig. 3. A 5×5 example of the DLS algorithm

Extended Deterministic Local Search

The DLS algorithm produces good results (see Sec. V), but it exhibits one important flaw: If none of the critical or worst points can be improved, the algorithm is terminated. While it is reasonable to terminate the algorithm if performance improvement is no longer possible, the DLS algorithm might produce a suboptimal, non-uniform design. In order to further improve the homogeneity of the data distribution, the DLS procedure is extended. A subtle but far-reaching alteration of the DLS algorithm leads to substantial improved results: Instead of stopping the algorithm after the nearest neighbor distance of the worst set of elements $\mathbb{W}$ can not be improved anymore, *all* elements are examined, see Algorithm 2. The

With the structure presented in algorithm 1, replace Step 2 with:
Step 2: Sort the vector $\vec{d}_{\mathrm{NN}}$ by increasing values. This forms a sorted set of points $\mathbb{W} \subset \mathbb{X}$.

Algorithm 2: Extended Deterministic Local search (EDLS)

critical distance does not necessarily improve in each iteration of the extended deterministic local search (EDLS) algorithm, but no deteriorations are accepted. With every successful iteration, the elements are distributed more homogeneously in the input space, thus converging to a uniform distribution. Also, by clearing space for other samples, the EDLS allows further iterations of formerly locked elements, thus the critical distance can further be improved during future iterations leading to a more homogeneous data distribution. Therefore

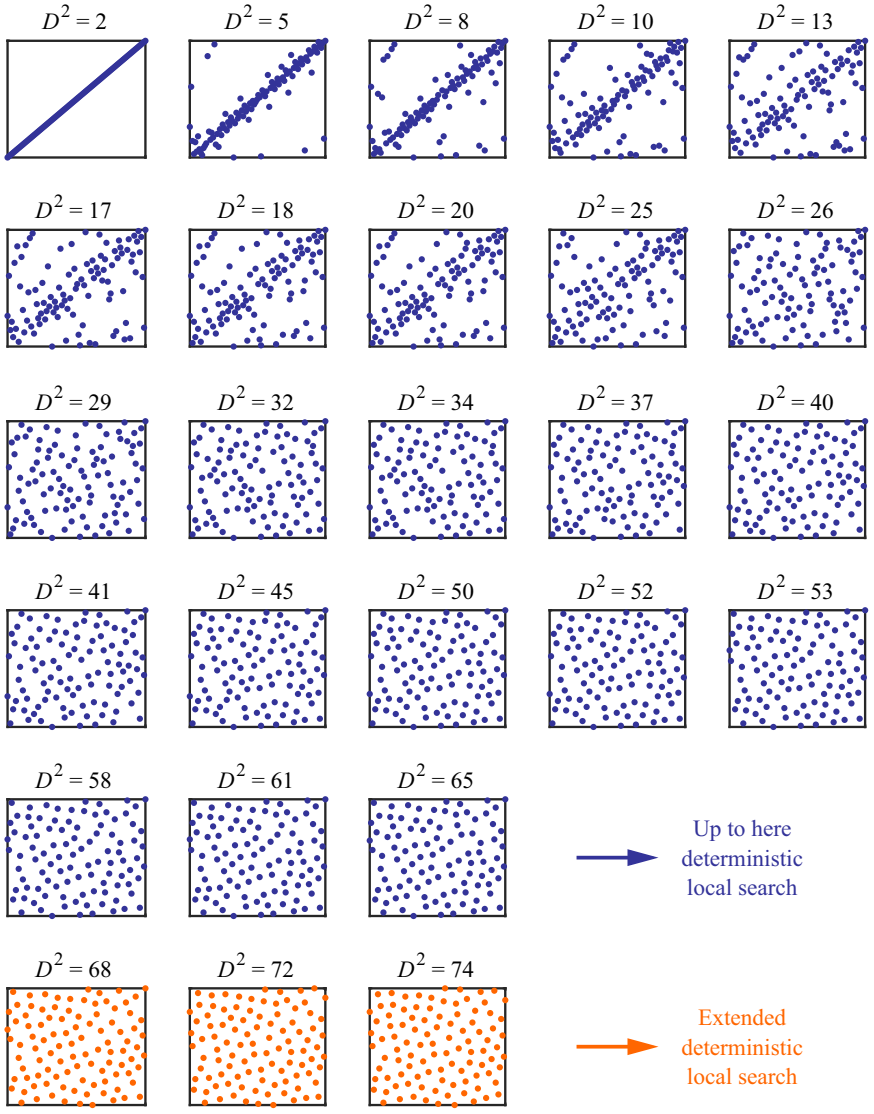


Fig. 4. Improvement of the D^2 values together with the corresponding data distribution for $n = 2$ inputs and $N = 100$ data points

the proposed Algorithm 2 improves the data distribution and thereby also makes room for and realizes further improvements in the critical distance (the actual optimization goal).

The complexity of the algorithm theoretically should be approximately quadratic in the number of LH data points because the number of pairs grows quadratically. Indeed this complexity can be experimentally verified. With respect to the input dimension an exponential complexity can be observed which seems to be a manifestation of the curse of dimensionality.

Further improvements of the algorithms:

The pseudo code of the algorithms above shows only a minimal realization of a local search. To improve the computation speed or quality of the results, further improvements are realized:

- It is not necessary to recalculate the distance to all neighbors in step 1 in every iteration. This is only necessary in the first iteration. The proximity relationship only change for a very limited number of samples in subsequent iterations, i.e. $\vec{x}_k$, $\vec{x}_j$, their former and new nearest neighbors.
- The set of coordinate exchange partner points $\mathbb{S}$ can be reduced by excluding useless for swapping samples. Samples who have been tested in earlier loop cycles within the same iteration must not be tested again. The set $\mathbb{S}$ becomes significantly smaller, as more elements of $\mathbb{W}$ are processed in Algorithm 2.
- The set of variables for coordinate exchange $\mathbb{V}$ should be permuted from one iteration to the next. This improves the quality of the result by impeding the alignment of samples

in repeating patterns. For many random initializations this is irrelevant, but the effect is visible in bad initializations, e.g. if all elements are diagonally aligned.

- It is possible to incorporate fixed elements. For example if an old set of measurements is available and should only be extended by some new measurements. The fixed elements must simply be excluded from $\mathbb{W}$ and $\mathbb{S}$, while remaining in $\mathbb{X}$. Though this may produce a collapsing design, the algorithm still improves the distribution of samples towards uniformity.
- If the number of critical point pairs after one swap decreases, the swap is still accepted. Though no improvement is visible in the D^2 value, the data will be more uniformly distributed. This adjustment of the swap acceptance improves the DLS as well as the EDLS algorithm. In the example from Fig. 4 the D^2 value of the DLS is improved from 52 to 65. The EDLS gets from a D^2 value of 72 to a value of 74.

V. Comparison

A comparison between our approach (EDLS) for the optimization of LH designs and two other algorithms is made, which are generally well-known for their good performance. One competitor is called translational propagation algorithm (TPLHD) and is based on geometrical patterns. The user must choose a so called seed design, that is translated into arbitrary high dimensions by recreating these patterns [9]. This heuristic procedure is very fast, but the achieved quality of the LH design highly depends on the chosen seed design. For this comparison we choose the 1 x n seed design as recommended by [9] and for which they claim to achieve comparable results to simulated annealing optimization approaches, at least up to six input variables. The second competitor to our approach is the iterated local search (ILS) algorithm for maximin LH designs proposed by [8]. In this procedure local optima found through local searches (see Algorithm 1) are perturbed by rearranging randomly chosen variables and points from the current design. Through the perturbation the local search is able to continue the optimization. Its result is only accepted, if the achieved LH design quality increases. Perturbations and subsequent local searches are repeated until a termination condition is met. In [8] it is shown, that the LH designs obtained by the ILS algorithm are superior to LH designs found by simulated annealing optimization approaches. In addition to the three competing approaches, the results from basic local search (DLS) are given as reference.

The comparison of the three algorithms is done for 2, 4 and 6 variables for 10 to 200 points. For a better readability of the following figures only every tenth point is plotted. The squared critical distance D^2 is plotted versus the number of points N. Comparison can only be done, if the algorithms have the same preconditions. Here we measure the calculation time required by our EDLS and use it as termination criterion for the ILS method to guarantee a good comparability. The calculation times reach from nearly one second ($N = 10$; $n = 2$) to two hours ($N = 200$; $n = 6$) on a simple personal computer. One convergence plot, belonging to the data distributions shown in Fig. 4, for $n = 2$ and $N = 100$ data points is exemplarily shown in Fig. 5. As can be seen, most of the calculation time

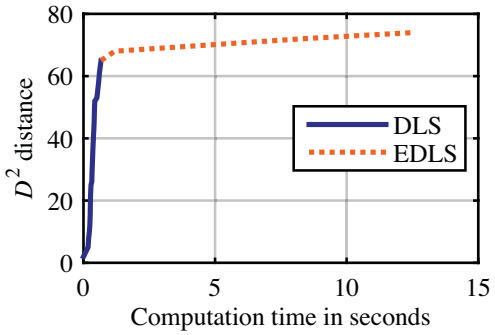


Fig. 5. Convergence plot of the D^2 distances obtained by the EDLS algorithm for $n = 2$ inputs and $N = 100$ data points

is needed for the extension of the original DLS algorithm. For large problems the algorithm can be stopped at any point in time because it improves continuously. Note that even though the D^2 distance does not significantly improve through the algorithm extension, a better uniformity of the data distribution is significant, as can be seen in Fig. 4. The result of the TPLHD algorithm and the intermediate result of our approach, i.e. the D^2 value after completing DLS without extension, are obtained much faster. The comparison is still legitimate, because both deterministic approaches have reached their best possible solution for each combination of points and variables. All search algorithms start with a diagonal LH design. This is the worst initialization for finding a space-filling design.

Figure 6 shows the experimental results for a design space consisting of 2 variables. Note that the D^2 values increase with the number of points N due to the fact, that the input scaling grows. Each axis covers values between $[1, N]$ and no normalization is done. This leads to a better interpretability of the achieved critical distances, since the minimum diagonal distance maintains to be n for an input space of dimension n. For a small number of points up to 70, all algorithms perform equally well. EDLS has similar D^2 values as the ILS algorithm, in some cases it is even slightly better. The TPLHD leads to better results in some particular cases, but is not generally superior. For a design space consisting of 4 variables shown in Fig. 7 the results are similar. For designs up to 80 Points the algorithms achieve similar results. With more points in the design space the TPLHD algorithm performs poorly and DLS without extension becomes successively worse. EDLS is slightly better than the ILS method and they both outperform the other two. By increasing the number of variables to 6 these effects are enhanced, as can be seen in Fig. 8. The quality of the LH designs is comparable up to 50 points. For larger number of points, the TPLHD algorithm becomes inferior. The ILS and EDLS approach are both superior, but EDLS is slightly better. With an increasing number of points DLS without extension becomes successively worse. Note that close

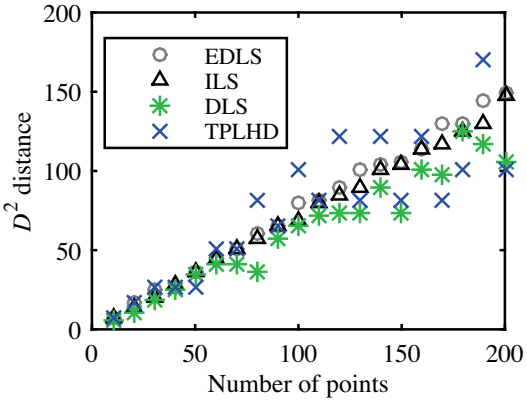


Fig. 6. D^2 distance for 10 to 200 Points for EDLS, ILS, DLS and TPLHD with 2 variables.

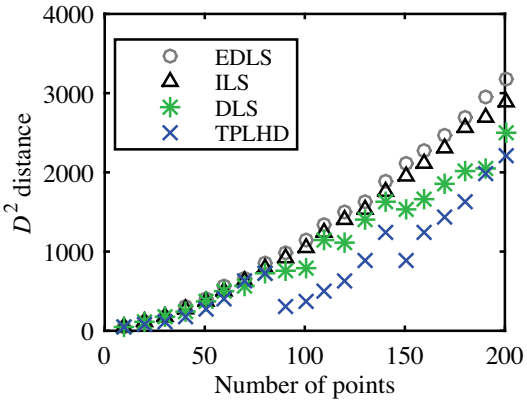


Fig. 7. D^2 distance for 10 to 200 Points for EDLS, ILS, DLS and TPLHD with 4 variables.

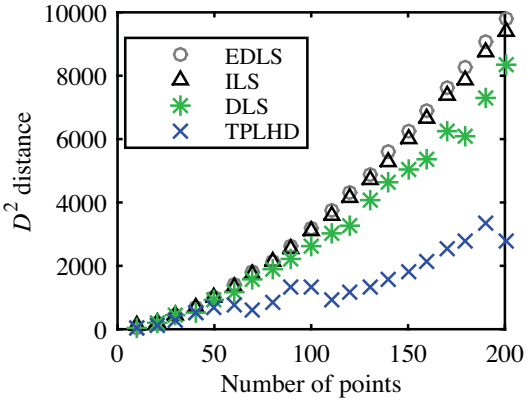
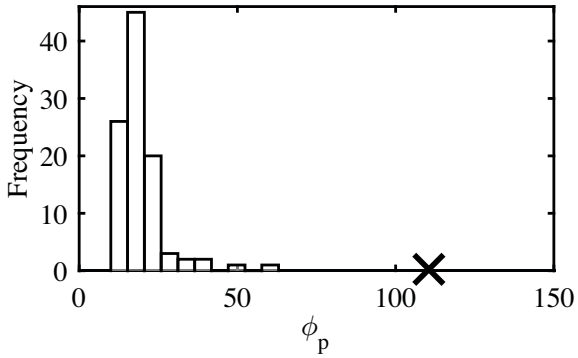


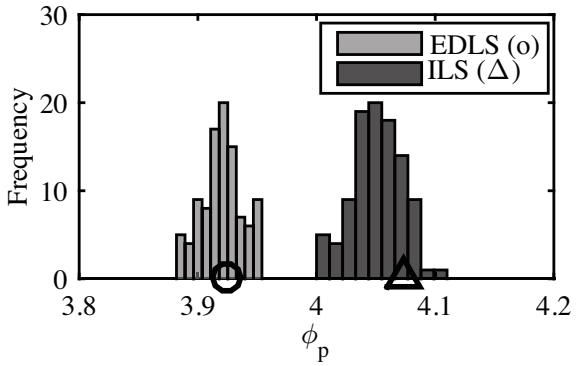
Fig. 8. D^2 distance for 10 to 200 Points for EDLS, ILS, DLS and TPLHD with 6 variables.

to the global optimum, small improvements in D^2 correspond to significantly improved point distributions.

Additionally, a comparison of the ILS method and our EDLS is done for different initial LH designs. Therefore 100 random LH designs are generated and are optimized afterwards with the two competing optimization algorithms. Again, the computation time needed by EDLS is used as the termination criterion for the ILS method. The comparison is performed for three different data amounts ($N = 50$; $N = 100$; $N = 200$) and $n = 2$, $n = 4$ as well as for $n = 6$ input variables. Figure 9 exemplarily shows the histograms for $N = 200$ data points and $n = 4$ input variables. Note that the bins count ϕ_p values, where smaller values indicate better data distributions. Instead of the squared critical distance D^2 the ϕ_p ($p = 50$) values are shown. Since all point-pair distances contribute to the ϕ_p value, it is a better representation of the whole data distribution. As can be seen from Fig. 9 the



(a) Initial designs



(b) Optimized designs

Fig. 9. Histograms of the initial (a) and optimized (b) ϕ_p values with the ILS method and EDLS for $N = 200$ and $n = 4$. The cross indicates the LH design, where all data points lay on the diagonal of the input space. The circle and triangle show the optimized designs.

initial designs vary strongly in their ϕ_p values (variance of initial ϕ_p values $\sigma^2_{ini} = 59.00$). The variance of the optimized LH designs is decreased heavily ($\sigma^2_{EDLS} = 2.85 \cdot 10^{-4}$ and $\sigma^2_{ILS} = 4.58 \cdot 10^{-4}$). The mean ϕ_p value achieved by EDLS is significantly smaller than the one obtained by using the

ILS algorithm for equal computation times, considering a significance level of $\alpha = 0.01$. This result is representative for all our performed calculations. The only exception, where no significant difference in the mean ϕ_p values can be found, is for $N = 50$ and $n = 2$, see Table I. The gap between the two optimization methods, that can be seen in Fig. 9b, tends to become bigger with an increasing number of points for all input dimensions. Besides the histograms for the randomly generated initial LH designs, values for the initialization on the diagonal of the hypercube are shown. Starting at the same ϕ_p value marked with a cross in Fig. 9a, the optimized values differ. Here EDLS yields a better result, indicated by the circle in Fig. 9b, than the ILS method, marked with a triangle in Fig. 9b.

TABLE I
MEAN ϕ_p VALUE ACHIEVED BY EDLS IS SIGNIFICANTLY BETTER (+) OR WORSE (−) THAN THE ONE ACHIEVED BY ILS ALGORITHM. NO SIGNIFICANT DIFFERENCE IS INDICATED BY O. THE SIGNIFICANCE LEVEL IS $\alpha = 0.01$.

$N =$ \ $n =$	2	4	6
50	o	+	+
100	+	+	+
200	+	+	+

In summary, for a small number of design points the algorithms yield comparable LH designs. For a two dimensional design space and very few data points no algorithm is generally superior. If the number of points increases for a given input dimensionality, EDLS outperforms the other approaches for the analyzed examples. For a practical use of these designs in engineering applications, often more design points are required to achieve a good result for modeling, especially for a high number of variables. Considering higher dimensional input spaces with an increasing number of points, EDLS tends to result in the best distributed designs.

VI. CONCLUSION

A new algorithm for the maximin optimization of LH designs is presented, that extends the local search used within the already existing ILS method proposed in [8]. It is based on element exchanges in the design matrix and pursues a greedy strategy in order to improve the uniformity of the data distribution. Our deterministic EDLS method is compared to two other algorithms, i.e. ILS [8] and TPLHD [9] as well as to an intermediate result, the DLS [8]. The investigations for different input space dimensionalities show the superiority of EDLS, if the number of samples is large. For small sample sizes all competitors achieve comparable results. Additionally the robustness of the EDLS as well as the ILS method with respect to varying initial LH designs is shown. The achieved uniformity of the data distribution for each algorithm scatters only slightly due to different initializations.

REFERENCES

[1] D. C. Montgomery, *Design and Analysis of Experiments*, 5th ed. John Wiley & Sons, Inc., USA, 2001.

[2] I. M. Sobol, "Distribution of points in a cube and approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, pp. 86–112, 1967.

[3] I. M. Sobol, D. Asotsky, A. Kreinin, and S. Kucherenko, "Construction and Comparisonof High-Dimensional Sobol' Generators," *Willmot Magazine*, pp. 64–79, Nov. 2011.

[4] R.-B. Chen, D.-N. Hsieh, Y. Hung, and W. Wang, "Optimizing latin hypercube designs by particle swarm," *Statistics and Computing*, vol. 23, no. 5, pp. 663–676, 2013.

[5] M. D. Morris and T. J. Mitchell, "Exploratory designs for computational experiments," *Journal of statistical planning and inference*, vol. 43, no. 3, pp. 381–402, 1995.

[6] R. Jin, W. Chen, and A. Sudjianto, "An efficient algorithm for constructing optimal design of computer experiments," *Journal of Statistical Planning and Inference*, vol. 134, no. 1, pp. 268–287, 2005.

[7] M. E. Johnson, L. M. Moore, and D. Ylvisaker, "Minimax and maximin distance designs," *Journal of statistical planning and inference*, vol. 26, no. 2, pp. 131–148, 1990.

[8] A. Grosso, A. Jamali, and M. Locatelli, "Finding maximin latin hypercube designs by iterated local search heuristics," *European Journal of Operational Research*, vol. 197, no. 2, pp. 541–547, 2009.

[9] F. A. C. Viana, G. Venter, and V. Balabanov, "An algorithm for fast optimal latin hypercube design of experiments," *International journal for numerical methods in engineering*, vol. 82, no. 2, pp. 135–156, 2010.

[10] B. Husslage, G. Rennen, E. R. Van Dam, and D. Den Hertog, *Space-filling Latin hypercube designs for computer experiments*. Tilburg University, 2006.

[11] F. A. C. Viana, "Things you wanted to know about the latin hypercube design and were afraid to ask," in *10th World Congress on Structural and Multidisciplinary Optimization, Orlando, Florida, USA (cf. p. 69)*, 2013.

[12] T. J. Santner, B. J. Williams, and W. Notz, *The design and analysis of computer experiments*. Springer, 2003.

[13] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[14] R. L. Iman and W. Conover, "Small sample sensitivity analysis techniques for computer models. with an application to risk assessment," *Communications in statistics-theory and methods*, vol. 9, no. 17, pp. 1749–1842, 1980.

[15] J.-S. Park, "Optimal latin-hypercube designs for computer experiments," *Journal of statistical planning and inference*, vol. 39, no. 1, pp. 95–111, 1994.