Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

UniDOE: An R Package for Uniform Design Construnction via Stochastic Optimization

Aijun Zhang
The University of Hong Kong

Haoyu Li
The University of Hong Kong

Abstract

This paper introduces a new R package called UniDOE for space-filling designs construction based on multiple choices of discrepancy criteria. We develop a stochastic optimization algorithm for exchanging factorial elements and setting adaptive thresholds innovatively. The algorithm may entertain flexible inputs of initial designs, including random, user specified or those with partial fixed columns. In the meanwhile, we also develop an algorithm for constructing augmented uniform designs, which searches for the best choice of augmented runs to an input uniform design. Both algorithms are implemented by C++ and built into the UniDOE package through RCpp interface. We demonstrate through examples that 1) for small scale settings, our construction may achieve lower criterion values than existing best designs to our knowledge, and 2) for large-scale settings, the UniDOE package may work well to generate approximately uniform designs. We provide a long list of new designs to replace the existing list in the website of the Uniform Design Association of China.

Keywords: Uniform design, descrepancy, stochastic optimization, threshold accepting, element exchange, run augmention, sequential experiment.

1. Introduction

Most projects in the world require building surrogate models based on computer generated experiments to reduce cost. It's a challenging task for both industries and researchers to devise an efficient design of experiment. To improve the space-filling property and computational efficiencies for designs, the uniform experimental design (Fang, 1980) is a prevalent method, which allocates experiments on the experimental domain in a uniform fashion (Fang, Ke and Elsawah, 2017). varieties of discrepancy measures such as centered-L2 discrepancy (CL2), mixture-L2 discrepancy (MD2) and maximin distance criterion are popularly used as objective functions.UniDOE is a package that aims at finding an efficient experimental design using a

stochastic evolutionary (SE) algorithm. The optimal experimental design problem is to search a design X in a given design class Z, target of which is to optimize it with respect to different criteria. For simplicity, Let $D(\cdot)$ be a discrepency function such as CL2 or MD2. Our package intends to seek for an optimal design $X^* \in Z$, which minimizes the objective function in Z, i.e.

$$D(X^*) = \min_{X \in Z} D(X)$$

There exist many stochastic searching algorithms that can help gradually decrease the discrepancy value in (1), such as TA and adjusted TA. TA algorithm contains an outer loop and an inner loop, which starts with an initial designX_0 generated from Z. In every iteration of inner loop, TA will decide whether to replace the old design with a new designX_{new}, which is made by manipulating some $element-wise\ exchanges$ in several columns to X_0. Specifically, the determination formula is

$$New\ Design = \begin{cases} X_0; & if\ \Delta D = D(X_{new}) - D(X_0) > T_i \\ X_{new}; & if\ \Delta D = D(X_{new}) - D(X_0) <= T_i \end{cases}$$

.T_i in the above formula is so-called *threshold*, which is a non-negative real number to determine the degree of hardness for changing X_0. In the outer-loop, a sequence of T = [T_0, T_1, ... , T_I] is pre-defined,which is decreasing to 0. This design of threshold allows the searching process to exit local minimum and the final design should reach a local minimum with good quality. For each threshold T_i in the outer loop, a pre-fixed number of iterations are executed so that the inner loop and outer loop can collaborate to improve the design. However, there exist some flaws in TA algorithm and sometimes it may lead to a worse design than X_0,i.e. $D(X_0) < D(X_{obj})$. In Fang *et al.*'s paper(2017), they proposeed an adjusted TA algorithm. In their paper, they first harness an mixture schedule to arrange threshold sequence. When T_i is large compared to T_0, it decreases in an exponential manner:

$$T_i = \frac{I - i}{I} T_{i-1}\ ,\ i = 2, ...I\ \&\ T_i > cT_0$$

On the other side, if T_i is too small, linear diminishing method will be adopted to preset threshold sequence T = [T_1, ..., T_I] as:

$$T_i = \frac{I = i - 1}{I - i} T_{i-1},\quad where\ i = 1, ..., I\&\ T_i <= cT_0$$

,where c is a constant ($0 < c < 1$). Following that, they illustrated their next procedures in $U(27, 3^{13})$.They first fix columns of initial design from online MA design $L_{27}(3^{13})$ to proceed TA algorithm. Then they use full factor level permutation to obtain a state-of-the-art design with $mixture\ L_2 - discrepancy(MD2) = 62.8011$.Though pre-defined sequence of threshold method in adjusted TA algorithm boosts the performance in searching process and find a better design in terms of $minimum\ abbration(MA)$ and $MD2$, it's not guaranteed that this interim threshold sequence will always be a good choice to step out from local optimum. In UniDOE, we utilize a stochastic adaptive algorithm in the search process, which adjusts threshold automatically and by following similar experiment procedures, this package leads to a new state-of-art in design $U(27, 3^{13})$, which achieves lowest mixture discrepancy according

to our knowledge with less computational time. The outline of this paper is as follows. In *section* 2, we provide elobarate dipiction about $Uniform\ Design\ Construction(UDC)$ function in UniDOE, which contains detailed algorithm, discrepancy criteria as well as selection of initial design. *Section* 3 will describe $Augmented\ UD\ Construction(AUDC)$ function in UniDOE,an augmented construction methodology for searching better designs. Furthermore, many numerical results will be expressed in *section* 4 and it exemplifies state-of-the-art work as well as some simple but useful usage of UniDOE.

2. Uniform Design Construction

Searching for experimental design can be proved to be an NP-hard problem, which is nearly computationally impossible when the size of design gets larger. This package exploits a statistical optimization algorithm which updates the design in an element-wise manner.

2.1. Discrepancy Criteria

Three most prevalent criteria are implemented in UniDOE package including Centralized L_2 discerpancy, Mixture L_2 Discrepancy and Maximin criteria. The first two criteria are utilized to construct uniform designs while the maximin criterion is used to build maximin distance designs, which aims at enlarging the distance between points.

Centralized L2 Discrepancy Criterion(CL2)

L_p is a measure of difference between empirical cumulative distribution function of an experimental design and uniform cumulative distribution function as shown below. It represents the non-uniformity of a design and a better design should have a lower discrepancy value.

$$D_p(P_n) = [\int_{C^s} |F_n(x) - F(x)|^p dx]^{\frac{1}{p}}$$

Compared to other L_p discrepancies, L_2 design can be expressed precisely with convenience of computation. Centralized L_2 discerpancy can be expressed more charmingly and satisfies $K - H\ inequality$ among three types of L_2 discrepancies proposed by Hickernell(1998) and it has a analytically precise formula:

$$CL_2(X)^2 = (\frac{13}{12})^s - \frac{2}{n}\sum_{i=1}^{n}\prod_{k=1}^{s}(1 + \frac{1}{2}|x_{ik} - 0.5| - \frac{1}{2}|x_{ik} - 0.5|^2)$$

$$+\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{s}(1 + \frac{1}{2}|x_{ik} - 0.5| + \frac{1}{2}|x_{jk} - 0.5| - \frac{1}{2}|x_{ik} - x_{jk}|)$$

,where n is runs, s is factors for the design.

Mixture L2 Discrepancy Criterion(CL2)

Recent research has shown that Centralized L_2 discerpancy contains some limitations when dealing with high dimensional data. The discrepancy value is less influenced by points near center of design. Repeated experiments will not provide additional information but sometimes it will have smaller discrepancy value.Under the motivation for finding a more efficient objective function, MD2 is formulized by Y.D.Zhou et al(2013), which satisfies following criteria(Fang et al,2006) for assessing measures of uniformity for construction of experimental designs.

- Measure uniformity of P on C^s.
- Measure projection uniformity of P on C^u, where u is a non-empty subset of 1,.,s.
- Invariant under permuting factors and runs.
- Invariant under coordinate rotation.
- Consistent with other criteria in experimental design.
- Geometric meaning.
- Fast and convenient for computation.
- Satisfy K-H inequality.

Formula for MD2 is given below:

$$MD^2(X) = (\frac{19}{12})^s - \frac{2}{n}\sum_{i=1}^{n}\prod_{j=1}^{s}(\frac{5}{3} - \frac{1}{4}|x_{ij} - \frac{1}{2}| - \frac{1}{4}|x_{ij} - \frac{1}{2}|^2)$$

$$+\frac{1}{n^2}\sum_{i=1}^{n}\sum_{k=1}^{n}\prod_{j=1}^{s}(\frac{15}{8} - \frac{1}{4}|x_{ij} - \frac{1}{2}| - \frac{1}{4}|x_{kj} - \frac{1}{2}| - \frac{3}{4}|x_{ij} - x_{kj}| + \frac{1}{2}|x_{ij} - x_{kj}|^2)$$

,where n is runs, s is factors for the design.

Maximin Criteria

Space-filling designs are crucial to constrain the disparity between sophisticated true model and computer generated approximate models. One important species of spacing-fillling design is Maximin distance design, which arranges points as even as possible among the space and seeks to maximize the minimum distances between points(i.e. rows of design). Maximin distance criteria was proposed by Johnson *et al*(1990) and it's asymptotically D-optimal under Bayesian setting, where D-optimal design of experiments is proven to be a NP-hard problem (Welch,1980).The inter-row distance is represented as :

$$\min_{1<=i,j<=n,i\neq j} d(x_i, x_j)$$

,Where d_{x_i,x_j} is distance between two sample points :

$$d(x_i, x_j) = [\sum_{k=1}^{s} |x_{ik} - x_{jk}|^m]^{\frac{1}{m}}$$

ϕ_p criterion was famously proposed by Morris and Mitchell (1995) as following:

$$\phi_p = (\sum_{i=2}^{n}\sum_{j=1}^{i-1}\frac{1}{d_{i,j}^p})^{\frac{1}{p}}$$

,in which $d_{i,j}$ is the discrepancy between i-th and j-th run of the experimental design and p is a positive integer. When p is large enough, ϕ_p criterion is equivalent to maximin distance criterion.

2.2. SOAT Algorithm

```
R> library(png)
R> library(grid)
R> library(gridExtra)
R> img1 <-  rasterGrob(as.raster(readPNG("./pictures/for_loop.png")), interpolate = FALSE)
R> img2 <-  rasterGrob(as.raster(readPNG("./pictures/while_loop_UpdateM.png")), interpolat
R> grid.arrange(img1, img2, ncol = 2)
```

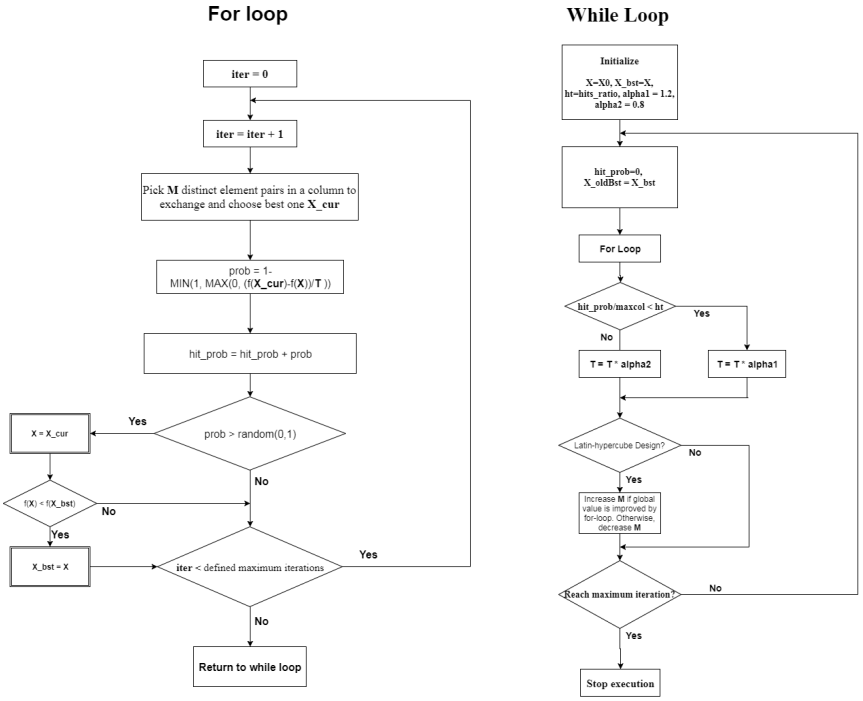

Figure 1 : *SOAT algorithm*

In **while-loop** of the *SOAT* algorithm, X_0 is the initial design, which can be randomly generated, user specified or those with partial fixed columns and complemented by UniDOE. *ht* is a probability ratio with default value=0.1. This parameter controls the threshold adjustment procedure. If ht<0.1, then threshold is enlarged so that more designs will be accepted in for-loop. Otherwise, threshold is decreased and less designs will be accepted in each for-loop.Threshold updating details will be elaborated in *section* 2.3. Another hyper-parameter M is updated following threshold change if the input design is latin-hypercube. M is increased if global criterion value is lowered in for-loop and decreased otherwise. Denote total number of possible entry-wise exchange of a column to be $N_p = \frac{n^2(q-1)}{2q}$, We also bound M in a range($0.2*N_p$, $0.3*N_p$) but less than 50 to guarantee its performance.

In **for-loop**, after initializing the threshold and design in while-loop, *SOAT* will proceed element exchanges for columns. In each column, it will randomly generate M distinct pairs of entries and calculate the discrepancy value correspondingly after swapping them. The M pairs of elements are changed without saving and finally we choose the pair with best discrepancy value to swap that leads to a new design X_{try}. The above procedure is similar to Ruichen Jin *et al*'s ESE algorithm(2005).

Next, *SOAT* utilized a probability measurement for determining whether to replace design X with X_{cur}. A value *prob* is set to be $1 - min(1, max(0, \frac{f(X_{cur})-f(x)}{T}))$ to represent the probability of replacing X. If a uniformly distributed variable produces value smaller than *prob*, then we update X to be X_{cur}. In essence, this experiment is a bernoulli trial with $P = prob$ with different *prob* in each iteration. We record the sum of expectations for all iteraions and denote it to be *hit_prob*. $ht = \frac{hit\_prob}{maxcol}$ is used to update threshold sequence as discussed in **while-loop** paragraph. If criterion value of X reaches a lower value than X_{best}, which means for-loop obtains a better global design. Then it replaces X_{best} with X and for-loop achieves improvement in this round. Accomplishing improvement or not decides the modification of M in while-loop.

2.3. Adaptive Threshold

Predefined threshold sequence in traditional TA algorithm is likely to lead the searching process into a bad local minimum value. This setting is inflexible and may result in a design even worse than initial design(Fang. K.T *et al*,2017. Though adjusted TA algorithm innovatively relieves the problem by using mixture set of threshold, the searching process still have limitations.To further boost the performance, we propose an **adaptive threshold updating** system in our **SOAT** algorithm and the result is satisfiable. At first, the T_0 is initialized to be a small proportion of discrepancy of initial design and then the threshold is updated according to different *ht*, which controls *hit_prob* ratio as shown in *section* 2.1. If *ht* is smaller than a value(default = 0.1), then threshold is increased so that more designs will be accepted in for-loop. Otherwise, threshold is decreased and less designs will be accepted in for-loop. The increasing and decreasing ratio are set to be 1.2 and 0.8 respectively, which were proven to be effective during our experiments. *ht* ratio should be kept at a small value in [0,1) so that only small proportion of designs with good quality can be accepted. *ht* is set as an argument in package functions and can be defined by users. Figure 2 gives a sample comparison between *Adjust TA* algorithm and *SOAT* algorithm in 10000 iterations on design $(20, 20^{19})$.

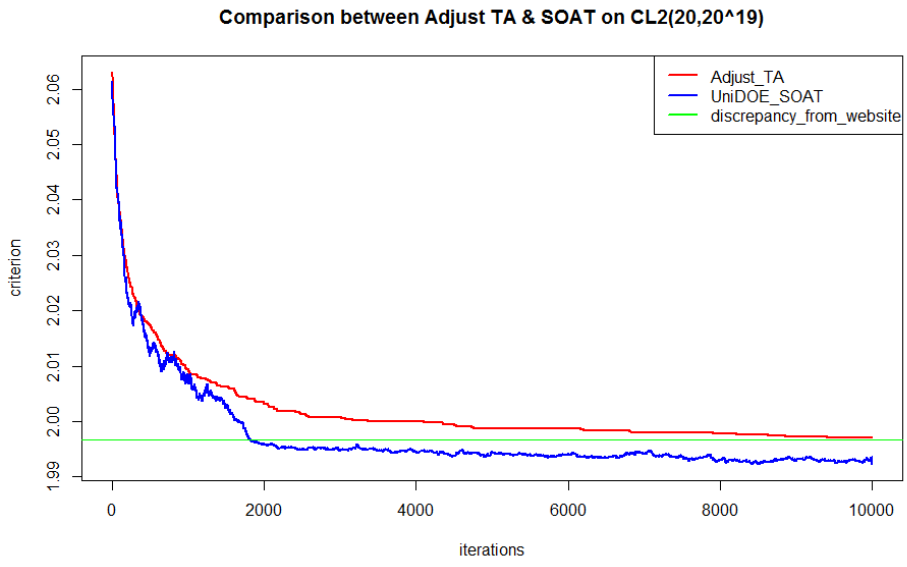

Figure 2 : *Comparison between adjusted−TA algorithm and SOAT algorithm on design* $(20, 20^{19})$

2.4. Element-wise exchange

There are *maxcol* iteraitons in inner loop. In each iteration, our algorithm performs M element-exchanges for distinct element pairs. Discrepancy values for each design are recorded the design with lowest discrepancy value is selected to be X_{try}. If $prob <= uniform\_random(0, 1)$, where $f(*)$ is the discrepancy function and $uniform\_random(0, 1)$ is generated from $(0, 1)$ uniformly, then X is changed to be X_{try} . The hidden logic behind it is that designs with punily greater discrepancy value should be more likely to be accepted. Designs with lower discrepancy value than $f(x_{best})$ is assured to be accepted. *prob* can formulate the relationship explicitly. Next, if $f(X) < f(X_{best})$, where $f(X_{best})$ is minimum discrepancy already reached, then we let $X_{best} = X$ and improvement is achieved in this round.

2.5. Initial Design Choice

There are three initial design choices in *UniDOE*:

Random initial design

UniDOE provides users with a simple and convenient manipulation to create an experimental design. Users only need to input runs(n), factors(s), levels(q) and discrepancy criteria("CL2", "MD2" or "maximin") to Uniform Design Function (UDC). Then the package will generate a random design with given parameters and pass it through **SOAT** algorithm to do optimization. For instance, if we want to obtain a design $D_{12}(3^4)$ with *Mixture* L_2 *Discrepancy* $(MD2)$ criterion, the user only needs to specify:

```
R> require(UniDOE)
R> n = 12
R> s = 3
R> q = 4
R> init = "rand" # randomly generated initial design, default choice.
R> crit = "MD2"
R> returnValue = UDC(n=n,s=s,q=q,init=init,crit=crit)
```

. The code above is succinct and the $returnValue$ is a list that contains initially generated design(Init_Matrix), optimized design(UniDOE_Matrix), initial discrepancy value(obj0), discrepancy value after searching(obj), time used to search(`time(s)`) and discrepancy list generated during the search(obj_list). The final design(transposed) can be retrieved through:

R> t(returnValue$UniDOE_Matrix)

User input design

Calculation of large-scale design is required and sometimes inevitable for many industries. Though the calculation speed is markedly accelerated by implementing main computation parts in C++, the time cost can still be high for very large designs. We achieved an option that allows users to input a already fine design and UniDOE will search for a better design based on this. There are mainly two reasons we do this:

- There already exist some good designs, such as tables in www.cms-ud.com/UD/UniformDesign.html. We can utilize them as initial input to save running time.

- Based on their past work and samples, users can test whether their own designs can be improved through *UniDOE*.

Partial fixed columns from MA design

Using partial fixed columns from MA matrix in initial design provides researchers a new perspective in constructing uniform designs. As shown in K.T.Fang (2017), they fixed columns from a traditionally constructed MA design $L_{27}(3^{13})$ and run adjusted-TA algorithm to decrease its discrepancy, after which, they perform all combination of level permutation to obtain a new benchmark for $D_{27}(3^{13})$. It suggests the potential for column-fixed methodology and this package also adopts this approach into consideration. In our experiments, we use last five columns of a tradition MA design (as shown in figure 3) into initial matrix and run UDC function to search for a better design. Then we use partial level permutation approach to further lower the discrepancy and we achieve the same discrepancy value($MD(U_{27}(3^{13})) = 82.8011$) as in K.T.Fang's paper(2017).

```
R> MA_design = as.matrix(read.csv("./OA_27_13_3.csv",header=FALSE))+1
R> knitr::kable(MA_design,format = "latex")
```

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	2	2	2	3	3	1	2	3	2	1	1
3	1	3	3	3	2	2	1	3	2	3	1	1
1	2	1	2	2	2	3	3	1	2	3	2	1
2	2	2	3	3	1	2	3	2	1	1	2	1
3	2	3	1	1	3	1	3	3	3	2	2	1
1	3	1	3	3	3	2	2	1	3	2	3	1
2	3	2	1	1	2	1	2	2	2	3	3	1
3	3	3	2	2	1	3	2	3	1	1	3	1
1	1	2	1	2	2	2	3	3	1	2	3	2
2	1	3	2	3	1	1	3	1	3	3	3	2
3	1	1	3	1	3	3	3	2	2	1	3	2
1	2	2	2	3	3	1	2	3	2	1	1	2
2	2	3	3	1	2	3	2	1	1	2	1	2
3	2	1	1	2	1	2	2	2	3	3	1	2
1	3	2	3	1	1	3	1	3	3	3	2	2
2	3	3	1	2	3	2	1	1	2	1	2	2
3	3	1	2	3	2	1	1	2	1	2	2	2
1	1	3	1	3	3	3	2	2	1	3	2	3
2	1	1	2	1	2	2	2	3	3	1	2	3
3	1	2	3	2	1	1	2	1	2	2	2	3
1	2	3	2	1	1	2	1	2	2	2	3	3
2	2	1	3	2	3	1	1	3	1	3	3	3
3	2	2	1	3	2	3	1	1	3	1	3	3
1	3	3	3	2	2	1	3	2	3	1	1	3
2	3	1	1	3	1	3	3	3	2	2	1	3
3	3	2	2	1	3	2	3	1	1	3	1	3

Figure 3 : *MA Design used in SOAT to obtain state − of − art*

Furthermore, we involve some 'randomness' ideology into our tests, and stochastically select 5 columns from a MA design. We fixed these columns and appended randomly created part to generate an initial design $D_{27}(3^{13})$. Then, by setting maxiter = 10000 with a few experiments within minutes or even seconds, our approach reaches a new state-of-the-art discrepancy $MD(U_{27}(3^{13})) = 62.7987$. The advanced design $D_{27}(3^{13})^T$ is shown in figure 4:

```
R> New_design = as.matrix(read.csv("./New_crit_27_3_13.csv",header=TRUE)[,c(-1)])
R> knitr::kable(New_design,format = "latex")
```

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	3	2	3	3	3	2	1	2	2	1	1	1
2	2	1	3	2	1	1	1	1	2	2	1	3
3	1	3	3	1	2	3	1	2	2	2	2	2
3	2	2	2	3	3	3	2	1	3	2	1	2
1	1	1	2	2	1	3	3	2	1	1	1	2
2	3	3	2	1	2	2	3	1	1	2	1	1
2	1	2	1	3	1	2	1	1	1	3	2	2
3	3	1	1	2	1	2	2	2	3	2	2	1
1	2	3	1	1	1	3	1	1	3	1	3	1
3	3	1	1	1	3	3	1	3	1	3	1	3
1	2	3	1	3	3	1	3	2	1	2	2	3
2	1	2	1	2	3	3	3	3	2	2	3	1
2	2	1	3	1	3	2	3	3	3	1	2	2
3	1	3	3	3	1	1	3	3	3	3	1	1
1	3	2	3	2	2	3	3	1	3	3	2	3
1	1	1	2	1	3	1	2	1	2	3	2	1
2	3	3	2	3	1	3	2	3	2	1	2	3
3	2	2	2	2	2	1	1	3	1	1	2	1
2	3	3	2	2	3	1	1	2	3	3	3	2
3	2	2	2	1	1	2	3	2	2	3	3	3
1	1	1	2	3	2	2	1	3	3	2	3	3
1	2	3	1	2	2	2	2	3	2	3	1	2
2	1	2	1	1	2	1	2	2	3	1	1	3
3	3	1	1	3	2	1	3	1	2	1	3	2
3	1	3	3	2	3	2	2	1	1	1	3	3
1	3	2	3	1	1	1	2	3	1	2	3	2
2	2	1	3	3	2	3	2	2	1	3	3	1

Figure 4 : *state − of − the − art* $U_{27}(3^{13})$ *in* $MD2$ *criterion*

Using partial fixed columns from a good design to continue optimization process is also called warm start methodology. Warm start is very important in design construction since it can both save computational time and improve the original criterion. In our experiments, the warm start method is actually the key point that we achieve the state-of-the-art result. However, it's not guaranteed that design with s-1 or s-2 columns are the best choice for further exploration. For instance, figure 5 illustrates the outcome by using warm start method to

find a better $MD_{27}(3^{13})$ in 30 rounds of each set of columns chosen from $C = [3, 4, \ ... \ , 12]$. The designs with randomly chosen s-1 and s-2 columns from $U_{UMA_{27}}$ do not have a better discrepancy value. But if we choose designs with s-7 or s-8 columns to be warm-start design, it gives us a promising performance.

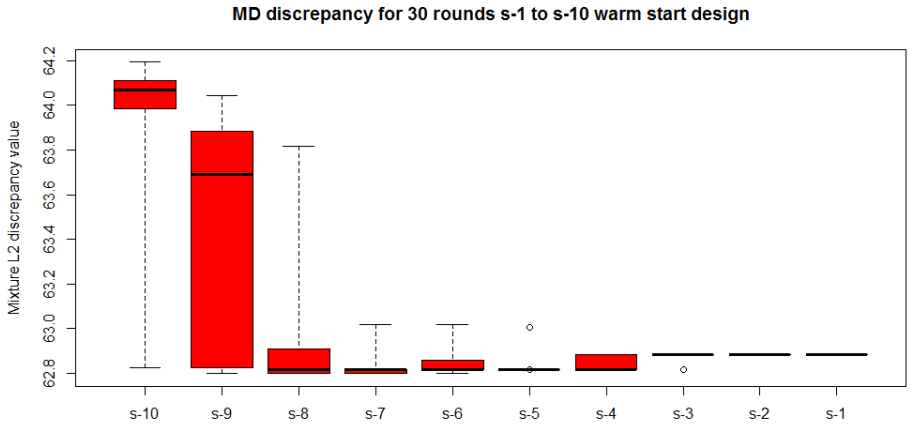

Figure 5 : *warm start designs experiments with* $s-1$ *to* $s-10$ *columns*

3. Augmented UD Construction

Augmented Uniform Design construction is another kind of warm start method for building uniform design. As suggested in Feng,Y. *et al*'s paper, by choosing appropriate discrepancy criterion such as $Wrap-around\ L_2-discrepancy\ criterion$ and warm-start designs, more flexible uniform design can be obtained through both row augmentation and column augmentation. In *UniDOE*, we also implemented row-augmented design construction method, which enables us to build design by adding runs. If an initial design is given as $D^{(1)}_{n_1}(q^s)$, then by augmenting n_2 runs to initial design, we can execute the optimizing algorithm with fixed n_1 runs to get a new uniform design $D^{(2)}_{n_1+n_2}(q^s)$.

3.1. Rowwise Strategy

Similar to warm start algorithm with fixed-column designs, which immobalizes given columns from existent design and updates added columns through element exchange, row augmented design only updates augmented rows and fix the given rows from initial design. However, to maintain the balanced property of design, we have to regulate some rules to the augmented rows and initial design. Let's define $nepl^I_x$ to be number of entries of a level at column x of initial design and the similar for $nepl^A_x$. n_I is number of runs in initial design and n_A is number of augmented rows. q_I and q_A are levels for initial design and augmented rows correspondingly. S_I to be number of columns of initial design and S_A to be number of columns of augmented rows. Then the following constraints should be satisfied:

$$nepl^I_x \ = \frac{n_I}{q} \ \forall x \in [1, \ ..., \ S]$$

$$nepl_x^A = \frac{n_A}{q} \; \forall x \in [1, \; ..., \; S]$$

$$S = S_I = S_A$$

$$q = q_I = q_A$$

In other words,

- Initial design must be balanced.
- Augmented rows must be balanced.
- Augmented rows must have identical number of columns as initial design.
- Initial design and augmented rows should have the same levels.

This paper will provide examples to elaborate usage of augmented uniform design construction(AUDC) function in section 4.

4. Numerical Results

In this section, we compare the performance of our new **UniDOE** package with another well-known package for design construction:**DiceDesign**, which implementented ESE(Enhanced stochastic evolutionary algorithm)in Ruichen Jin's paper(2005). Then we will illustrate the utilization of **UniDOE** for both research and industrial purpose.

4.1. Comparison with DiceDesign

Construction for large-scale uniform design is always stuck by using traditional R packages since the calculation time is often formidable. The motivation for combining the benefits of compiling language(like C or C++) and R creates **UniDOE**,which utilizes the power of both C++ and R. The following example illustrates the computational time and criterion by using pure R language, e.g. **DiceDesign** and mixed languages(R&C++)**UniDOE** to calculate latin hyper cube design $D_{20}(20^{19})$ based on $CL2$ using respective algorithms in two packages:

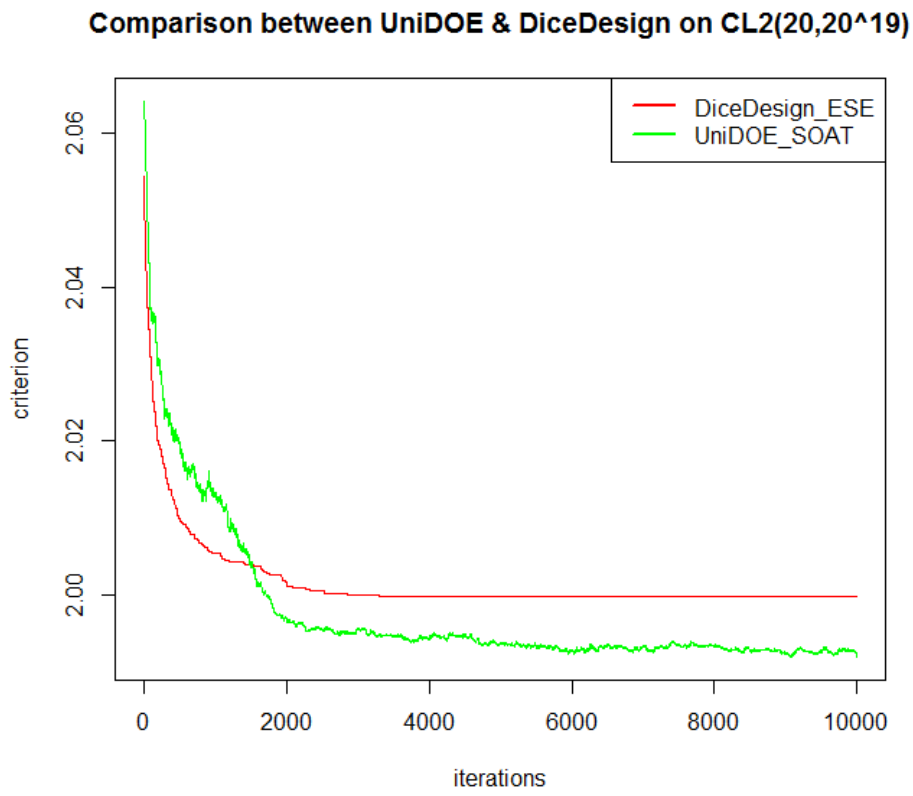

Figure 6 : *Criterion Comparison between UniDOE and DiceDesign* $D_{20}(20^{19})$ *in CL2 criterion*

Both ESE algorithm in **DiceDesign** and our SOAT algorithm start with an randomly generated design. In ESE algorithm, the parameter setting is inner_it = 100, J=50 and it=100. In SOAT algorithm, the total iteration is 10k times which equals to inner_it*it and J(in our case, M) is adaptively changed as discussed in *section* 2.1. As shown in *Figure* 6, the final criterion value by running SOAT is better than ESE algorithm from DiceDesign. The reason why SOAT curve fluctuates up and down credits to the adaptive sequence of threhsold T and inner loop iteration M. With unfixed threshold sequence T and M, Design is more likely to jump out from local minimum criterion concave.

4.2. Element-wise augmentation

As discussed in section 3.1, rowwise augmentation is also implemented in UniDOE package. With a fixed uniform design as input, augmented design construction(AUDC) could obtain better results in some situations. As shown in *Figure* 7, using nearly the same time consumption, row-wise augmented design $D(60+60, 6^{30})$ generates a lower discrepancy value than searches from randomly initialized design. Assume we have an asymptotic uniform design $D_0 = U(n, q^s)$ and we want to optimize a design $D_{aug}(n+n_1, q^s)$. Running AUDC function to generate augmented design is illustrated as below. Users need to provide a good initial design D0 and number of expanded rows n1.

```
R> crit="CL2" # user input criterion.
R> D_aug = AUDC(X0=D0,n=n1,crit=crit)
```

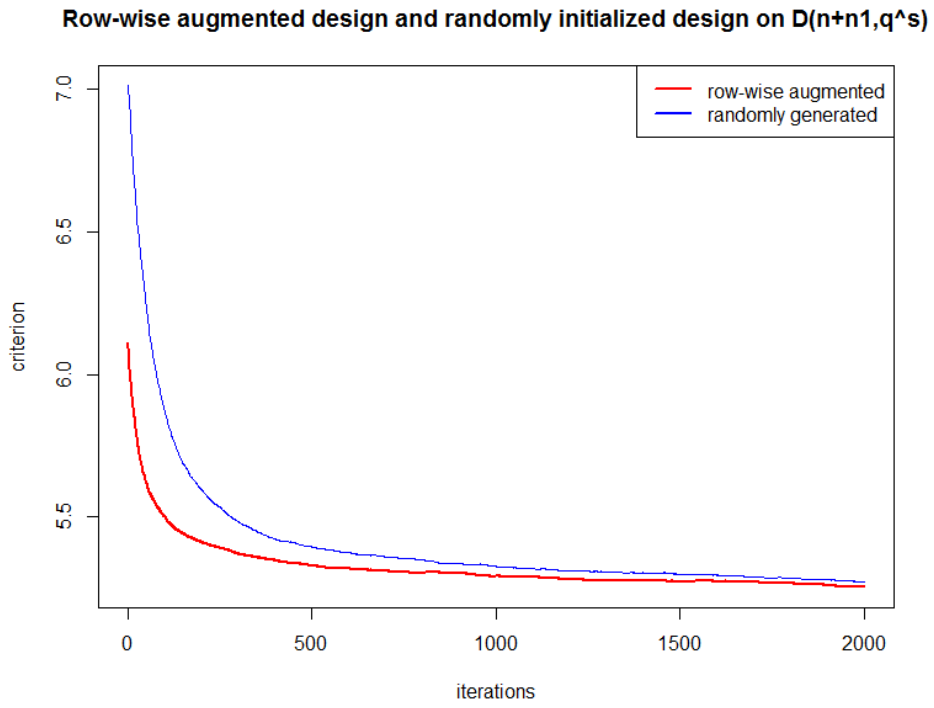

Figure 7 : $D(n+n_1, q^s)$ Comparison between row−wise augmentation and randomly initialized design

4.3. Refresh the UD Website Catelogue

By using our **UniDOE** package, we updated the online best design tables from website of Uniform Design Association of China: www.cms-ud.com/UD/UniformDesign.html with rarely computational burden. The average improvement ratio for all Latin Hyper Cube designs approximates 85%, and 60% for general designs. After utilizing the efficient algorithm in **UniDOE**, we have also provided more relatively large-scale designs to compensate for the leaks in industry. More details for the advanced designs are appended in appendix and are available online through: https://github.com/HAOYU-LI/UniDOE/tree/master/Advanced_Designs.

5. Discussion and Future work

In this paper, we introduce a Stochastic optimization updating (SOAT) algorithm for efficient construction of uniform designs. The algorithm is composed of for-loop and while-loop. In while-loop, we provide an adaptive methodology for updating thresholds. Threshold is used to accept new designs in for-loop. We further adaptively update parameter M used in for-loop to help latin-hypercube designs jump out from local minimum value if there is no improvement in last round. M is also used to help designs to converge faster to a new lower local minimum value, which improves the searching efficiency. In for-loop, we abstract the acceptance problem into a bernoulli trial problem, which has different probability $P = prob$ at each inner iteration. Through sum of their expectations, we control the threshold updating process in while-loop. After experiments, our package achieves a new state-of-the-art value on the design D(27,3^{13})

in Mixture Discrepancy criterion. And our package saves tens to hundreds times of time consumption compared with other existent algorithms and packages in our knowledge with better criterion optimization performance. Comparisons have been made in *section* 2.3 *and* 4.1. Furthermore, we use our package to update the online best design table from website of Uniform Design Association of China at a high improvement ratio. For the convenience of future research, **UniDOE** also implements rowwise augmented design construction method through AUDC function, which is the prevelant trend for experimental design field.

References

1. Fang, K.T. (1980). The uniform designs: application of number-theoretic methods in experimental design. *Acta Math. Appl. Sinica*, **3**, 363-372.

2. Fang, K.T., Dennis K. J. Lin, Winker, P., & Zhang, Y. (2000). Uniform Design: Theory and Application. Technometrics, **42(3)**, 237-248. doi:10.2307/1271079

3. Fang, K.T. Li, R.Z. A. Sudjianto.(2006) Design and modeling for computer experiments, Chapman and Hall/CRC, New York.

4. Fang, K.T., Ke X. and Elsawah, A.M. (2017). Construction of uniform designs via an adjusted threshold accepting algorithm. *Journal Of Complexity*, ...

5. Hickernell, F. J. (1998). A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, **67**, 299-322.

6. Jin, R.C., Chen, W. and Sudjianto, A. (2005). An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference*, **134**(1), 268-287.

7. Johnson, M. and Moore, L. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, **26**, 131-148.

8. Ke, X., Zhang, R. and Ye, H.J. (2015). Two- and three-level lower bounds for mixture L2-discrepancy and construction of uniform designs by threshold accepting. *Journal of Complexity*, **31**(5), 741-753.

9. Morris, M. D., & Mitchell, T. J. (1995). Exploratory designs for computational experiments. Journal of statistical planning and inference, 43(3), 381-402.

10. Welch,J.W. (1980). Algorithmic complexity: three NP- hard problems in computational statistics. Taylor & Francis. 17-25.

11. Yang, Zhou, & Zhang. (2017). Augmented uniform designs. Journal of Statistical Planning and Inference, 182, 61-73.

12. Zhou, Y.D., Fang, K.T. and Ning, J.H. (2013). Mixture discrepancy for quasi-random point sets. *Journal of Complexity*, **29**, 283-301.

Affiliation:

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd