Chapter 6
The Construction of Optimal Design for Order-of-Addition Experiment via Threshold Accepting

Peter Winker, Jianbin Chen, and Dennis K. J. Lin

Abstract The objective of the order-of-addition (OofA) experiment is to find the optimal addition order by comparing all responses with different orders. Assuming that the OofA experiment involves $m(\geq 2)$ components, there are $m!$ different orders of adding sequence. When m is large, it is infeasible to compare all $m!$ possible solutions (for example, $10! \approx 3.6$ millions). Two potential construction methods are systematic combinatorial construction and computer algorithmic search. Computer search methods presented in the literature for constructing optimal fractional designs of OofA experiments appear rather simplistic. In this paper, based on the pairwise-order (PWO) model and the tapered PWO model, the threshold accepting algorithm is applied to construct the optimal design (D-efficiency for the present application) with subsets of size n among all possible size $m!$. In practical, the designs obtained by threshold accepting algorithm for $4 \leq m \leq 30$ with $n = m(m-1)/2 + 1, m(m-1) + 1, 3m(m-1)/2 + 1$ respectively are provided for practical uses. This is apparently the most complete list of order-of-addition (OofA) designs via computer search for $4 \leq m \leq 30$ in the literature. Their efficiencies are illustrated by a scheduling problem.

Keywords D-optimal design · Pair-wise ordering (pwo) mode · Threshold accepting · Tapered pwo model

P. Winker
Justus-Liebig-University Giessen, Licher Str. 64, 35394 Giessen, Germany
e-mail: Peter.Winker@wi.jlug.de

J. Chen
School of Statistics and Data Science & LPMC, Nankai University, Tianjin 300071, China
e-mail: chenjianbinlzu@163.com

D. K. J. Lin (✉)
Department of Statistics, Purdue University, 250 N. University Street, West Lafayette, IN 47907, USA
e-mail: dkjlin@purdue.edu

J. Fan and J. Pan (eds.), *Contemporary Experimental Design, Multivariate Analysis and Data Mining*,
https://doi.org/10.1007/978-3-030-46161-4_6

6.1 Introduction

The order-of-addition (OofA) experiment has been popularly used when the response of interest is affected by the addition sequence of materials or components. Considering the addition of m different materials or components into the system, the different responses depend on different adding orders. Each permutation of $\{1, \dots, m\}$ is a possible adding order, hence there are $m!$ different orders of adding sequences into the system which yield different responses. The OofA experiments are prevalent in many scientific and industrial areas, such as chemistry-related areas, bio-chemistry, food science, nutritional science, and pharmaceutical science.

The purpose of OofA experiment is to find the optimal addition order by comparing all possible responses with different orders. However, it is often infeasible to test all the $m!$ possible orders when m is large (for example, 10! is about 3.6 millions). In practice, a number of randomly selected orders are tested, but the empirical experience indicates that randomly selected orders may not be most informative. Hence the design problem arises to choose a subset of orders for comparison. A good design for the OofA experiments will help experimenters to identify the important order effects, and to find out the optimal addition order with substantially fewer experimental runs. Such an important problem has received a great deal of attention in the past decades. For example [18] considered the design with pair-wise ordering (PWO) effects. Based on the PWO model [19] proposed a number of design criteria and found some OofA designs which have the same correlation structures as the full OofA designs, for small number of components (m). Peng et al. [17] considered different types of optimality criteria and discussed the properties of some fractional designs. Zhao et al. [25] considered the minimal-point OofA designs. Yang et al. [24] has obtained a number of OofA designs called component orthogonal arrays (COAs) that are optimal under their component-position model. [13] reviewed the latest work on the design and model of OofA experiments, and introduced some new thoughts. [1] proposed another type OofA design named pair-wise ordering distance (PWOD) arrays that can be used in any models in the literature. Chen et al. [2] introduced a statistical method to speculate solutions of NP-hard problem involving orders by making use of design for OofA experiment.

This paper makes use of the threshold accepting algorithm to find the best subset of size n which implies searching for the optimal value of the objective function among all $m!$. This threshold accepting?algorithm provides high quality approximations to the global optimum. Therefore, designs obtained by our algorithm, involving only a fraction of all $m!$ possible permutations of components, are powerful for fitting models in terms of the D-efficiency and are efficient for predicting the optimal order. An illustrative example is provided to show the advantages of the obtained designs.

The remaining part of this article is organized as follows. Section 6.2 introduces the PWO model, the tapered PWO model and some optimality criteria. The threshold accepting algorithm is proposed in Sect. 6.3. The optimal fractional OofA designs obtained by the threshold algorithm are provided in Sect. 6.4, and a scheduling example is discussed in Sect. 6.5. Section 6.6 gives some concluding remarks.

6.2 Preliminary

6.2.1 PWO Model

The order-of-addition (OofA) experiment involves $m(\geq 2)$ components, and there are $m!$ different orders of adding sequences into the system to yield different responses. For any pair of components i and j, if the impact of component i preceding j is different from the impact of component j preceding i, such a difference is called the effect of pair (i, j). To express the order effect [18] proposed "pseudo factor". [19] called it pair-wise ordering (PWO) factor. The PWO factor is defined as

$$I_{i,j} = \begin{cases} +1 & if \;\; i \;\; precedes \;\; j, \\ -1 & if \;\; j \;\; precedes \;\; i. \end{cases} \tag{6.1}$$

This indicates whether the component i precedes the component j or not, where i and j are the components. There are $q = \binom{m}{2}$ PWO factors, corresponding to all pairs of component orders. These factors are arranged according to the lexicographic ordering of the components' indices. For illustration, when $m = 4$ and a possible order $\pi = 2143$ is given, we have $I_{12}(\pi) = -1, I_{13}(\pi) = +1, I_{14}(\pi) = +1, I_{23}(\pi) = +1, I_{24}(\pi) = +1$ and $I_{34}(\pi) = -1$. Assuming β_{ij} is the effect to response caused by I_{ij}, the PWO model is the first-order model by summing the effects of all I_{ij}'s, namely:

$$y = \beta_0 + \sum_{i<j} \beta_{ij} I_{ij} + \varepsilon, \tag{6.2}$$

where y is the response of interest, ε is a random error assumed to be independent and to have a normal distribution $N(0, \sigma^2)$, and $p = q + 1$ parameters $\{\beta_0, \beta_{12}, \beta_{13}, \ldots, \beta_{(m-1)m}\}$ should be estimated.

In practice, it is not affordable to test all the $m!$ orders when m is large. Let $\pi = (\pi_1, \ldots, \pi_m)$ be a permutation of $\{1, \ldots, m\}$ which specifies the order. Denote Π as the subset of size n from all of $m!$ possible orders. Based on the best subset Π, the expected PWO model can be written as

$$E(y|\pi) = \beta_0 + \sum_{i<j} \beta_{ij} I_{ij}(\pi), \qquad \pi \in \Pi. \tag{6.3}$$

The PWO model has two merits. Firstly, it is easy to interpret: the effect of β_{ij} shows the difference between the impacts of all the possible orders in which i precedes j and the impacts of all the orders in which j precedes i. Secondly, the PWO model is an economic model which requires a small number of runs ($p = q + 1$) compared with the total number of candidate runs ($m!$). In order to fit PWO model (6.3) for using the smallest runs [25] proposed a special type of design with only $q + 1$ runs (out of

$m!$ runs), which is called a minimal-point OofA design, as long as its D-efficiency is nonzero.

6.2.2 *Tapered PWO Model*

Although the PWO model is an economical model, the PWO effect has some weaknesses, for example, the PWO effect I_{12} in the sequences "$1 \to 2 \to \ldots$", "$1 \to \ldots \to 2$", "$\ldots \to 1 \to 2 \ldots$" and "$\ldots \to 1 \to \ldots \to 2 \ldots$" is assumed to be the same ($I_{12} = +1$). Obviously, these sequences have different sense between the component 1 and the component 2. It is possible to assume that the impact of any such pairwise order changes with an increase in the distance between the components in the pair in practice. So another model of interest is "tapered PWO model" as considered as in [17].

For any components $i, j (i \neq j)$ and $\pi = (\pi_1, \ldots, \pi_m) \in \Pi$, let $h(ij, \pi)$ be the distance between i and j in π, i.e., $h(ij, \pi) = |k - l|$ if $\pi_k = i$ and $\pi_l = j$, so that $h(ij, \pi) \in \{1, \ldots, m-1\}$. Denote

$$z_{ij} = \begin{cases} c_{h(ij,\pi)} & \textit{if } \ i \ \textit{ precedes } \ j \ \textit{ in } \ \pi \\ -c_{h(ij,\pi)} & \textit{if } \ j \ \textit{ precedes } \ i \ \textit{ in } \ \pi \end{cases} \tag{6.4}$$

as the "tapered PWO factor", where $c_h = 1/h$ or $c_h = c^{h-1}$ with known $c\,(0 < c < 1)$ for $h \in \{1, \ldots, m-1\}$. Then, the tapered PWO model can be expressed as

$$y = \beta_0 + \sum_{i<j} \beta_{ij} z_{ij} + \varepsilon, \tag{6.5}$$

where y is the response of interest, ε is a random error assumed to be independent and to have a normal distribution $N(0, \sigma^2)$, and β_0 and β_{ij} are the unknown parameters. For any $\pi = (\pi_1, \ldots, \pi_m) \in \Pi$, the corresponding tapered PWO model can be represented as

$$E(y|\pi) = \beta_0 + \sum_{i<j} \beta_{ij} z_{ij}(\pi), \qquad \pi \in \Pi. \tag{6.6}$$

The tapered PWO model is a generalized PWO model, if one chooses $c_h = 1$ for all h, then the tapering PWO factor (6.4) and the tapered PWO model (6.5) respectively become the PWO factor (6.1) and the usual PWO model (6.2) of [18] and [19].

Under the PWO model or the tapered model, this paper makes use of the threshold accepting algorithm to construct the optimal design, for all subsets of size $n\,(\geq q + 1)$ among all possible $m!$ order. For simplicity, three particular run sizes of OofA designs are of interest, even though the proposed algorithm is capable for any n. Recall that $q = \binom{m}{2}$.

(a) Minimal-point design: $n = q + 1$,
(b) Double type design: $n = 2 * q + 1$,
(c) Triplicate type design: $n = 3 * q + 1$.

6.2.3 *Some Optimality Criteria*

There are many criteria for constructing optimal designs in literature. Let X denotes the model matrix with respect to the pre-specified model, p is the number of columns of X and n be the run size of X, then, the D-value of a design D is defined by $D_e(D) = \frac{1}{n}|X^TX|^{\frac{1}{p}}$, which is proportional to the generalized variance of the parameter estimators $\widehat{\beta}$ to be minimized. That is, the volume of the confidence ellipsoid for $\widehat{\beta}$ is minimized by maximizing the determinant $det(X^TX)$. A-optimal designs are those designs which minimize the average variance of the estimators $\widehat{\beta}$ and thus the criterion $trace((X^TX)^{-1})$. The E-criterion focus on the minimum eigenvalue of X^TX. We will select all possible choices of designs, by considering which one(s) attain the optimum in terms of these criteria. There are more other optimal design criteria (see, for example [16]).

In this paper, we mainly focus on the D-efficiency under the pre-specified model (PWO model or tapered PWO model) for simplicity. Likewise, the $A-$, $E-$efficiency can be defined. The larger D-efficiency the better, an optimal design has a D-efficiency of 1. Throughout this paper, let D_{full} denote the D-efficiency of the full design. For all other designs, the relative D-efficiency $D_r = D_e/D_{full}$ is used here. For the tapered PWO model, let $q = \binom{m}{2} = m(m-1)/2$ and $p = q + 1$ [17] proposed the D-efficiency of the design D is

$$D_r(D) = \frac{D_e(D)}{\left[\{b_0 + (m-2)b_1\}^{m-1}(b_0 - 2b_1)^{(m-1)(m-2)/2}\right]^{1/p}}, \tag{6.7}$$

where

$$b_0 = 2\{(m-1)c_1^2 + \Lambda + c_{m-1}^2\}/\{m(m-1)\},$$

$$b_1 = 2\Sigma_h\{m - h(1) - h(2)\}c_{h_{(1)}}\{2c_{h_{(1)}+h_{(2)}} - c_{h_{(2)}}\}/\{m(m-1)(m-2)\},$$

and Σ_h denotes the sum over positive integers $h(1), h(2)$ such that $h(1) + h(2) \leq m - 1$. For the PWO model, $c_h = 1$ for all h, $b_0 = 1$ and $b_1 = 1/3$, then the D-efficiency of the design D (6.7) reduces to

$$D_r(D) = D_e(D)/\left[\frac{(m+1)^{m-1}}{3^q}\right]^{1/p}. \tag{6.8}$$

Our goal here is to maximize the objective functions given in (6.7) and (6.8), making use of the threshold accepting algorithm as described below.

6.3 The Threshold Accepting Algorithm

The problem of finding good OofA designs might be interpreted as a complex discrete optimization problem. For a given number of m components, the full design matrix comprises $m!$ rows. For a given objective function, selecting the best subset of size n implies searching for the largest value of the objective function among all subsets of size n of a set of size $m!$, i.e. in a discrete set of size

$$S = \binom{m!}{n} .$$

It is obvious, that a full enumeration of this set is beyond available computational ressources except for very modest values of m and n.

In related problems of finding optimal U-type designs, the use of stochastic local search heuristics turned out to provide high quality results, which for some instances with theoretical lower bounds could be shown to be globally optimal [8]. Therefore, it appears appropriate to follow a similar strategy for the problem of finding good OofA designs. Specifically, we use an implementation of the threshold accepting heuristic [3], which is a simplified version of simulated annealing by using a deterministic acceptance criterion for each local move. It also belongs to the class of local search methods sequentially moving through the search space by making small changes to a given design. When comparing a new design with the current one, it allows downhill moves, i.e. accepts solutions which are (slightly) worse than the previous one, in order to escape local maxima. By decreasing the threshold, up to which a worsening of the objective function is allowed in a search step, to zero over the run time of the algorithm, the algorithm provides high quality approximations to the global optimum.

A survey on different heuristic approaches which could be used for the present problem instance can be found in [23], and a detailed description of the threshold accepting algorithm with several applications including some in experimental design is provided by [20]. Previous applications in the context of experimental design include the calculation of lower bounds for the star-discrepancy [21], and the generation of low discrepancy U-type designs for the star-discrepancy [22], several modifications of the L_2-discrepancy [5], for CL_2 [4, 7], and for CL_2 and WL_2 [6, 9]. Furthermore, further details on low-discrepancy designs can be found in [10, 12, 14]

The pseudo code for the threshold accepting implementation used for the OofA design problem is exhibited in Algorithm 1. Thereby, D stands for the D-criterion to be maximized. The algorithm remains unchanged if instead of D another objective function has to be maximized. For an objective function to be minimized, the algorithm can be applied on minus this objective function. In the results section, we will report the values of the D-criterion relative to the theoretical maximum for the full PWO design, i.e. the relative D-efficiency D_r.

The threshold accepting algorithm conducts a local search strategy on the set of all OofA designs with m components and n runs denoted by $\mathscr{O}(m, n)$. The initial

Algorithm 1 Pseudo-code for Threshold Accepting

```
1: Initialize $n_R$, $n_{S_r}$ and the sequence of thresholds $\tau_r$, $r = 1, 2, \ldots, n_R$
2: Generate starting design $O^0 \in \mathscr{O}(m, n)$
3: for $r = 1$ to $n_R$ do
4:     for $i = 1$ to $n_{S_r}$ do
5:         Generate $O^1 \in \mathscr{N}(O^0)$    (neighbor of $O^0$)
6:         if $D(O^1) > D(O^0) - \tau_r$ then
7:             $O^0 = O^1$
8:         end if
9:     end for
10: end for
```

design O^0 is selected randomly (2:). It should be noted that selecting a "good" initial design, which might correspond to a local maximum of the objective function does not improve the performance of the algorithm. Instead, using the best out of some repeated runs of the algorithm for different randomly selected initial designs might result in an improved performance and higher robustness as compared to a single run with a corresponding larger number of iterations.

Starting from the initial design, a local search step is repeated a substantial number of times. In each search step, a new candidate design O^1 is selected randomly within a neighborhood of the current design $\mathscr{N}(O^0) \subset \mathscr{O}(m, n)$ (5:). The value of the objective function for the new candidate solution is calculated $D(O^1)$. If it turns out to be larger than the one of the current design O^0, it will be accepted and becomes the current design (7:). However, the new design will also be accepted if it is worse than the current one, though only up to a certain threshold defined by the current value of the threshold sequence (τ_r) (6:). This "threshold accepting" behavior of the algorithm avoids getting stuck in suboptimal local maxima of the objective function. Nevertheless, as the sequence of threshold values decreases to zero during the course of the algorithm, towards the end of a run, only improvements will be accepted. The current implementation uses an elitist approach, i.e., the best design obtained during the runtime of the algorithm is reported. For a properly tuned implementation of the algorithm, this should be equal to or at least quite close to the last design accepted by the algorithm.

While the overall layout of the algorithm is simple, and it proofed to be robust to minor modifications of neighbourhood structure and parameter settings, the actual performance still depends on some problem specific tuning. The three most important aspects are the choice of local neighbourhoods, the specification of the threshold sequence, and, for obvious reasons, the total number of iterations the local search step is repeated within the algorithm. With regard to the definition of neighbourhoods, we follow the experience from earlier applcations of threshold accepting in experimental design. Starting with a design O^0, a small number of rows (2 in our implementation) are randomly selected. These rows are exchanged with a row differing only in the ordering of few elements close to each other. In principle, this definition of local neighbourhood also allows for a fast updating of the some objective functions as described in [6] for the first time. However, for the current implementation this feature is not implemented yet. Nevertheless, given the tremendous growth in computational resources available, it is feasible to conduct repeated runs (10) for each problem instance with up to 10 000 000 iterations. In the results section, we report the best result obtained over all these runs. The corresponding designs are provided in the appendix.

The sequence of threshold values τ_r, $r = 1, \ldots, n_R$ is generated according to a data driven procedure first described in [20]. The rational of the approach is the observations that when performing local search over a discrete and finite search space such as $\mathscr{O}(m, n)$, also local changes of the objective function can take on only a finite number of different values. For the threshold accepting steps, all values of τ_r falling between two actual occurring differences will result in the same decision. Therefore, the threshold sequence is obtained by an empirical approximation to the underlying distribution of local changes of the objective function. To this end, first, a large number of OofA designs are randomly generated. For each of these designs a random neighbor is obtained employing the definition of local neighborhoods introduced before. The absolute value of the difference of the objective function between the two designs is calculated. The values are sorted in decreasing order and—given that too large thresholds imply an almost random search process—a lower quantile of these sequence is used as the actual sequence of threshold values. For the current application, the 60% of lower values (including zeros if the neighbor selected happens to be identical to the original design) is used.

6.4 Main Results

The best designs obtained by the threshold accepting algorithm are presented in Tables 6.1 and 6.2 (for $m \leq 30$). Both tables report the number of components m, the number of runs n, the D-efficiency as compared to the full PWO or full tapered PWO design, and the relation of n to the number of runs of the full designs. The corresponding designs are provided online (www.jlug.de/optimaloofadesigns). Given that there does not exist a closed form expression for the D-criterion of tapered PWO designs, we report D-efficiency for tapered PWO designs only up to $m = 10$. For larger values of m, the straightforward calculation of the D-criterion fails due to memory constraints. Therefore, Table 6.2 provides only results for the PWO designs, although using the algorithm also tapered PWO designs can be obtained for $m > 10$. A further constraint is imposed due to the numerical precision when calculating the values of the objective function. Using standard double precision only values of m up to about 30 are feasible, but the algorithm could be adjusted to work with higher precision.

Table 6.1 *D*-efficiency of tapering OofA designs obtained by threshold accepting

Components (m)	Runs (n)	TA optimized design		Share of runs ($n/m!$)
		D-efficiency PWO	*D*-efficiency Tapered PWO	
4	7	0.89613	0.84433	0.2917
4	13	0.98571	0.98585	0.5417
4	19	0.98122	0.98097	0.7917
5	11	0.90267	0.91904	0.0917
5	21	0.97278	0.97848	0.1750
5	31	0.98733	0.98974	0.2583
6	16	0.88107	0.84169	0.0222
6	31	0.97039	0.96663	0.0431
6	46	0.98854	0.98629	0.0649
7	22	0.81196	0.77259	4.3651×10^{-3}
7	43	0.96517	0.95798	8.5317×10^{-3}
7	64	0.98285	0.98217	12.6984×10^{-3}
8	29	0.75717	0.73876	0.7192×10^{-3}
8	57	0.95166	0.94345	1.4137×10^{-3}
8	85	0.97750	0.97429	2.1081×10^{-3}
9	37	0.72626	0.69174	0.1020×10^{-3}
9	73	0.93923	0.93100	0.2012×10^{-3}
9	109	0.97339	0.96662	0.3004×10^{-3}
10	46	0.68087	0.65436	0.0127×10^{-3}
10	91	0.92463	0.91838	0.0251×10^{-3}
10	136	0.96336	0.95770	0.0375×10^{-3}

Note: For each m, the D-efficiency of the obtained design for three experimental runs n are displayed: (a) the minimal point n = q + 1; (b) double type n = 2q + 1; (c) triplicate type n = 3q + 1; where $q = \binom{m}{2}$

The results indicate that with a rather small number of runs, highly efficient designs can be obtained. For Case (a), the minimal-point design $n = q + 1$; all designs reach at least 80% of the efficiency of the full design, though with only a small fraction of runs, especially for large m, $n/m!$ becomes almost zero—a substantial saving. For example, for $m = 20$ and $n = 191$, we have $n/m! = 0.0785 \times 10^{-15}$. Hence, if the practitioner attempts to save resource and time, the minimal-point design is a good choice. For Case (b), the double type design $n = 2q + 1$; all designs reach at least 95% of the efficiency of the full design, while the corresponding $n/m!$ is also almost zero. For example, for $m = 20$ and $n = 381$, we have $n/m! = 0.1566 \times 10^{-15}$. We recommend the experimenter who seeks designs with high D-efficiency to use the double designs when the resource and time allow. For Case (c), the triplicate type design $n = 3q + 1$; the optimal designs attain at least 97% of the efficiency of the full design, while the run size $n/m!$ is near zero. For example, for $m = 20$ and

Table 6.2 *D*-efficiency of OofA designs obtained by threshold accepting for $m > 10$

Components (m)	Runs (n)	TA optimized design D-efficiency (PWO)	Share of runs ($n/m!$)
11	56	0.80170	1.4029×10^{-6}
11	111	0.95969	2.7808×10^{-6}
11	166	0.98228	14.1586×10^{-6}
12	67	0.78958	0.1399×10^{-6}
12	133	0.95646	0.2777×10^{-6}
12	199	0.98081	0.4154×10^{-6}
13	79	0.77952	0.0127×10^{-6}
13	157	0.95238	0.0252×10^{-6}
13	235	0.97934	0.0377×10^{-6}
14	92	0.76463	1.0553×10^{-9}
14	183	0.94925	2.0991×10^{-9}
14	274	0.97744	3.1430×10^{-9}
15	106	0.75398	0.0811×10^{-9}
15	211	0.94704	0.1614×10^{-9}
15	316	0.97637	10.2417×10^{-9}
16	121	0.74091	0.0058×10^{-9}
16	241	0.94420	0.0115×10^{-9}
16	361	0.97389	0.0173×10^{-9}
17	137	0.73361	0.3852×10^{-12}
17	273	0.94096	0.7675×10^{-12}
17	409	0.97229	1.1499×10^{-12}
18	154	0.72681	0.0241×10^{-12}
18	307	0.93764	0.0480×10^{-12}
18	460	0.97088	0.0718×10^{-12}
19	172	0.71426	1.4139×10^{-15}
19	343	0.93483	2.8197×10^{-15}
19	514	0.96900	14.2254×10^{-15}
20	191	0.70542	0.0785×10^{-15}
20	381	0.93160	0.1566×10^{-15}
20	571	0.96728	0.2347×10^{-15}
25	301	0.65850	1.9405×10^{-23}
25	601	0.91783	3.8746×10^{-23}
25	901	0.95955	5.8087×10^{-23}
30*	871	0.90459	3.2837×10^{-30}
30	1306	0.95064	4.9236×10^{-30}

Note 1: For each m, the D-efficiency of the obtained design for three experimental runs n are displayed: (a) the minimal point n = q + 1; (b) double type n = 2q + 1; (c) triplicate type n = 3q + 1; where $q = \binom{m}{2}$.
Note 2: Using standard double precision only values of m up to about 30 are feasible (for m = 30 it depends on n), but the algorithm could be adjusted to work with higher precision

$n = 571$, we have $n/m! = 0.2347 \times 10^{-15}$. The triplicate designs are recommended due to a high D-efficiency, if there are no restrictions on the experimental conditions. In general, all the obtained three experimental designs with a small number of runs ($n = q + 1, 2q + 1, 3q + 1$) achieve a high level of D-efficiency when the underlying model is the PWO or the tapered PWO model.

6.5 Example: Scheduling Problem

The purpose of the scheduling problem is to schedule the resources and tasks to be optimized with regard to one or more objectives. A popular class of scheduling problem is the "job scheduling" problem, which seeks an optimal order of these jobs. Consider m jobs requiring processing in a certain machine environment, the scheduler hopes to sequence these jobs under given constraints. Let p_i $(i = 1, \ldots, m)$ represents the processing time of job i on a machine, ω_i being pre-specified weights, which reflects the importance of job i relative to the other jobs in the system. A schedule $\pi = (\pi_1, \ldots, \pi_m)$ is a permutation of $\{1, \ldots, m\}$ which specifies the order in which to process jobs. The completion time of the operation of job j is denoted as $C_j(\pi) = \sum_{i=1}^{j} p_i$, and the total cost function (total weighted completion time) is denoted by $W = \sum_{k=1}^{m} W_k = \sum_{k=1}^{m} \omega_k C_k^2$, where W_k denotes the cost function of job k. The objective is to find a optimal job-order such that the total cost function W is minimized.

For illustration, a simple example of job scheduling for a single machine model with $m = 3$ is discussed. Suppose the processing times are 5, 3 and 2 h for the 1st job, 2nd job and 3rd job, respectively, and the weights (cost coefficients) of these jobs are 6, 8, 7, respectively. Consider the job-order $1 \to 2 \to 3$, the completion time for job 1 is $C_1 = p_1 = 5hr$, the completion time for job 2 is $C_2 = p_1 + p_2 = 5 + 3 = 8hr$, the completion time for job 3 is $C_3 = p_1 + p_2 + p_3 = 5 + 3 + 2 = 10hr$. Thus the job-order $1 \to 2 \to 3$ has a total cost function: $W = 6 \times 5^2 + 8 \times 8^2 + 7 \times 10^2 = 1362$.

The purpose of the job scheduling problem is to find the optimal sequence from all possible solutions to minimize the total penalty. This is the same as the goal of the OofA problem. Hence we can consider the job scheduling problem as an OofA experiments problem. The PWO model can be used as the approximate model to deal with the job scheduling problem. If we have the prior information of the cost function, we can compute the costs of all possible job orders and compare all possible different orders to learn the dependence of the response on the order. For the illustrative example, since there is only $m! = 3! = 6$ potential orders, one can evaluate all possible order to find the optimal order. However, with m components to add, an exhaustive search of all permutations requires $m!$ runs of experiments, which is usually not affordable. So the design problem arises to choose a subset of orders for comparison.

Next, we discuss a case that 10 jobs are to be sequenced on a single machine with quadratic penalty function of its completion time. The pre-specified weights and processing times of these jobs are randomly generated from χ_1^2, such as $p = (0.1451700, 0.7428453, 7.1142859, 1.8774267, 7.1185982, 1.1431286, 10.5172882, 2.1484186, 2.4950454, 2.8989094)$, $\omega = (2.2094712, 7.1116628, 0.4190265, 6.7777317, 1.2965368, 1.5379331, 0.7221195, 1.7368003, 0.5205548, 0.3908880)$. With 10 components, it is infeasible to conduct all possible orders (tests) $10! \approx 3.6$ million. This is especially true for physical experiments and some expensive computer experiments. To fit the PWO model for using the smallest runs to save costs, we use the minimal-point OofA design $n = \binom{10}{2} + 1 = 46$ runs obtained from our algorithm in this paper. Under the quadratic penalty function, the 46 runs design and the corresponding total cost are shown in Table 6.3. Upon using the least squares approach, the parameters $\widehat{\beta}_{ij}$ in PWO model (6.2) are estimated. Ultimately, an OofA experiment is to find the optimal addition order. According to the method proposed by [2], the active variables are very critical for selecting the optimal orders. Considering all of degree of freedom for the minimal-point design are used to estimate the parameters in PWO model, there is no remaining degree of freedom to estimate the standard deviation. The Lenth ([11]) method is used here to identify the active variables, we take the pseudo standard error (PSE) to estimate the standard deviation. Upon calculating the Lenth statistics: $t_{PSE,i} = \widehat{\beta}_{ij}/PSE$, the active variables are tabulated in Table 6.4.

The favorable pattern "i precedes j" indicates an edge from i to j. From the favorable patterns of all active variables, one can always generate the corresponding directed graph by sequentially considering the significant parameters according to the absolute values of the active variables' estimated effects. In other words, we first consider the directed edge determined by the most significant parameter (the largest effect), then consider the directed edge from the second significant parameter (the second largest effect), and so on. In this procedure, we omit the active variables that are contrary to the current generated directed graph. Take the sequence '1628' as example, we sequently consider the active variables $\widehat{\beta}_{2,8}$ (966.425), $\widehat{\beta}_{2,6}$ (775.033), $\widehat{\beta}_{1,6}$ (700.148), $\widehat{\beta}_{1,8}$ (624.832). In this way, we omit the active variable $\widehat{\beta}_{1,8}$, because it is contrary to the generated sequence '1628'.

From the favorable patterns exhibited in Table 6.4, the directed graph of 10 jobs results in Fig. 6.1. According to the Fig. 6.1, the first five components of an optimal sequence are '16284', and the remaining optimal possible job orders can be obtained by permutating the last components '3, 5, 9, 10, 7', hence the possible numbers of order is $5!/2 = 120$. One of possible optimal orders is $1 \longrightarrow 6 \longrightarrow 2 \longrightarrow 8 \longrightarrow 4 \longrightarrow 5 \longrightarrow 9 \longrightarrow 10 \longrightarrow 7 \longrightarrow 3$, and the cost function is 1958.716.

We randomly select 100 runs from 10! and compute the costs of these 100 runs. As a comparison, we randomly select 54 (we already run 46 experiments to collect the data, for fairness, $100 - 46 = 54$ possible runs are selected) possible orders from our possible 120 orders, the results are showed in Fig. 6.2a. It can be seen that the costs of our method are smaller than the costs of randomly selected. This highlights the merit of our approach.

Table 6.3 The job-orders and total costs for a 10 job scheduling problem

Run	Order										Cost (W)
1	4	5	10	2	3	8	9	6	7	1	7463.671
2	8	3	4	10	6	5	9	2	7	1	10754.214
3	3	1	8	10	6	2	9	5	7	4	12674.186
4	9	6	4	10	8	7	3	5	2	1	14862.981
5	8	2	10	6	4	7	9	1	3	5	4318.170
6	9	2	7	3	6	5	10	1	8	4	15859.299
7	3	9	1	7	8	5	4	10	2	6	20486.508
8	1	5	6	2	9	3	10	4	7	8	8041.683
9	6	10	2	4	3	9	1	8	7	5	4193.411
10	10	6	7	2	1	8	5	4	3	9	9654.549
11	8	9	7	5	1	10	3	4	2	6	21368.549
12	10	8	2	6	4	3	5	7	1	9	5686.498
13	10	2	1	7	8	5	9	3	6	4	12976.774
14	7	4	9	1	3	10	6	5	8	2	16184.010
15	5	6	8	2	1	4	10	9	3	7	4205.401
16	9	3	7	8	6	4	2	1	5	10	14663.133
17	7	1	6	4	2	8	3	5	10	9	6363.397
18	8	5	9	2	10	1	3	6	7	4	12750.318
19	8	6	4	9	3	10	1	2	7	5	5912.879
20	3	8	2	7	1	4	5	6	9	10	9265.334
21	10	4	1	3	2	6	9	7	5	8	6175.202
22	4	2	8	1	7	3	9	10	5	6	4791.054
23	4	6	7	8	2	9	5	3	10	1	7447.334
24	10	3	7	6	1	9	5	4	2	8	21755.562
25	5	2	9	8	4	3	1	6	10	7	5465.729
26	3	6	8	1	10	7	4	5	2	9	15743.304
27	3	5	8	9	4	10	1	7	6	2	17428.615
28	10	9	2	7	4	6	3	8	5	1	9745.608
29	7	10	5	8	3	9	1	2	6	4	23354.098
30	1	6	3	5	4	2	7	10	9	8	8598.878
31	10	3	7	4	1	8	9	5	6	2	19101.778
32	4	6	1	3	10	8	9	5	7	2	11780.016
33	3	9	4	1	2	5	10	6	8	7	5984.161
34	6	1	5	7	8	9	10	3	2	4	19423.341
35	1	3	10	7	2	4	9	5	8	6	13212.515
36	1	3	7	2	6	8	4	9	10	5	10300.234
37	8	2	9	3	7	4	10	6	1	5	9889.003
38	4	9	6	8	10	7	1	2	5	3	6620.735
39	7	10	2	8	3	1	6	4	9	5	11364.447
40	10	5	7	8	6	4	3	2	9	1	18716.247
41	9	5	6	3	8	7	10	4	2	1	22834.160
42	8	6	7	4	5	10	1	2	3	9	10426.000
43	1	4	8	6	9	7	2	10	3	5	5203.967
44	7	8	9	1	4	6	10	2	3	5	9175.293
45	9	3	8	5	6	7	10	1	2	4	22223.030
46	10	8	1	9	4	7	6	5	3	2	12630.479

Table 6.4 The active variables for 10 jobs scheduling

Active variables	Estimator of the effects	Signs of the effects	Favorable patterns
$I_{1,2}$	−442.110	−	1 precedes 2
$I_{1,6}$	−700.148	−	1 precedes 6
$I_{1,8}$	624.832	+	8 precedes 1
$I_{1,10}$	−492.119	−	1 precedes 10
$I_{2,3}$	−1110.545	−	2 precedes 3
$I_{2,4}$	−431.188	−	2 precedes 4
$I_{2,5}$	−1314.436	−	2 precedes 5
$I_{2,6}$	775.033	+	6 precedes 2
$I_{2,7}$	−1349.35	−	2 precedes 7
$I_{2,8}$	−966.425	−	2 precedes 8
$I_{2,10}$	−858.088	−	2 precedes 10
$I_{3,4}$	631.513	+	4 precedes 3
$I_{3,6}$	476.540	+	6 precedes 3
$I_{3,10}$	452.552	+	10 precedes 3
$I_{4,5}$	−1343.049	−	4 precedes 5
$I_{4,7}$	−562.938	−	4 precedes 7
$I_{4,8}$	785.005	+	8 precedes 4
$I_{4,9}$	−714.446	−	4 precedes 9
$I_{4,10}$	−1039.873	−	4 precedes 10
$I_{6,7}$	−674.048	−	6 precedes 7
$I_{7,10}$	641.382	+	10 precedes 7

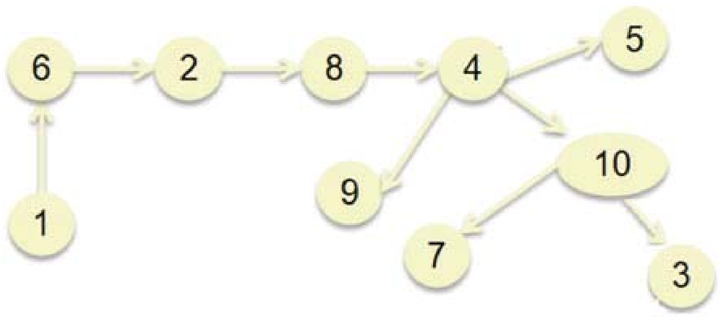
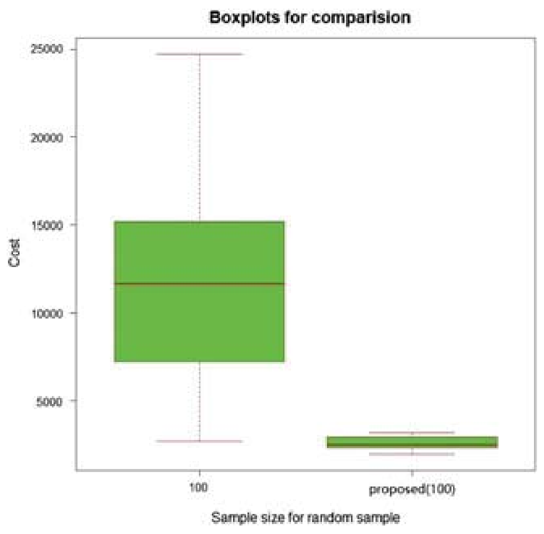
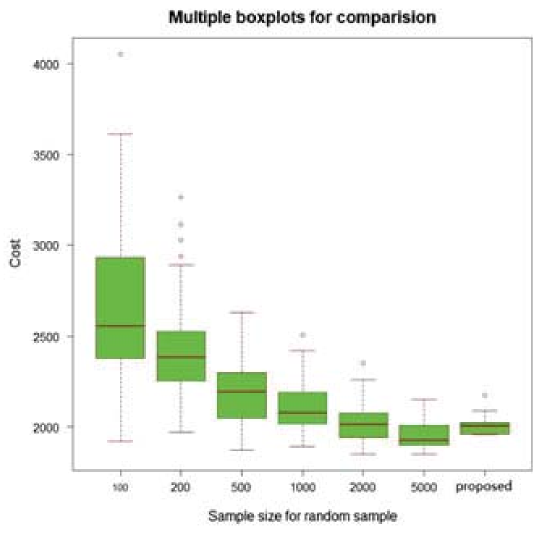


Fig. 6.1 Directed Graph



(a) 100 from 10! and 120 possible orders



(b) different sample sizes

Fig. 6.2 Boxplots of different sample sizes

As a comparison benchmark, a sample of $n = 100$ orders was randomly chosen (from all possible 10! orders) and their corresponding costs were evaluated. Among these 100 costs, the minimal cost was recorded. We repeat this process for 50 times. These 50 minimal costs were displayed as the first boxplot in Fig. 6.2b. Similarly, the same process is applied to $n = 200$, 500, 1000, 2000 and 5000 respectively. Their boxplots for minimal costs are also displayed in Fig. 6.2b. As expected, the larger n, the smaller minimal cost is found (with more consistency as well). As compared to our finding, we randomly chose 54 orders (out of the obtained result of 120 orders). Their corresponding costs were evaluated and the minimal cost is recorded. We also repeat this process for 50 times and the results is displayed as the last boxplot in Fig. 6.2b. It is clear that the performance of the proposed method with 100 ($= 46 + 54$) runs is as good as the random sample with 5000 runs. This more or less confirms the validity of the proposed method.

6.6 Conclusion

Order-of-addition (OofA) experiments have increasingly received a great deal of attention in scientific and industrial applications. This paper uses threshold accepting algorithm to provide a class of minimal-point designs with $n = q + 1$ runs, a class of double type design with $n = 2q + 1$ runs and a class of triplicate type design with $n = 3q + 1$ runs under the PWO model and tapered PWO model, respectively. The D-efficiency of these designs are shown in Tables 6.1 and 6.2. As a matter of fact, the threshold accepting algorithm can be used to construct the optimal design among any size n under any other models, such as, the triplet-order model [15] and the component-position model [24].

It is shown that a threshold accepting implementation can be used to generate highly efficient designs for OofA experiments. Here, the analysis was restricted to consider PWO designs. However, the framework is general enough to allow the construction of efficient designs taking into account higher orders. For improving the performance of the algorithm for larger values of m and n, the implementation has to be modified both with regard to the generation of random OofA designs and their mapping to the corresponding Z-matrices and with regard to the updating of the objective function for small modifications of a design in a neighbourhood. These improvements of the algorithm will allow tackling also larger problem instances and higher orders regarding the sequence of additions.

Acknowledgements All designs obtained can be found in website: www.jlug.de/optimaloofadesigns. This work was partially supported by the *National Science Foundation* via Grant DMS 18102925. The work of Jianbin Chen was supported by the National Natural Science Foundation of China (Grant Nos. 11771220). Professor Kai-Tai Fang has been a true leader in our society and has been a strong supporter of young fellows. His original work on uniform design had a significant impact on this work. It is our great privilege to contribute this work on computer experiments to this special issue in honor of his 80th birthday.

References

1. Chen, J.B., Han, X.X., Yang, L.Q., Ge, G.N., Zhou, Y.D.: Fractional designs for order of addition experiments. Submitt. Manuscr. (2019)
2. Chen, J.B., Peng, J.Y., Lin, D.K.J.: A statistical perspective on NP-Hard problems: making uses of design for order-of-addition experiment. Manuscript (2019)
3. Dueck, G., Scheuer, T.: Threshold accepting: a general purpose algorithm appearing superior to simulated annealing. J. Comput. Phys. **90**, 161–175 (1990)
4. Fang, K.-T., Lin, D.K.J.: Uniform experimental designs and their applications in industry. In: Khattree, R., Rao, C.R. (eds.) Handbook of Statistics, vol. 22, pp. 131–170. Elsevier, Amsterdam (2003)
5. Fang, K.-T., Lin, D.K.J., Winker, P., Zhang, Y.: Uniform design: theory and application. Technometrics **42**, 237–248 (2000)
6. Fang, K.-T., Lu, X., Winker, P.: Lower bounds for centered and wrap-around L_2-discrepancies and construction of uniform designs by Threshold Accepting. J. Complex **19**, 692–711 (2003)
7. Fang, K.-T., Ma, C.-X., Winker, P.: Centered L_2 discrepancy of random sampling and latin hypercube design, and construction of uniform designs. Math. Comput. **71**, 275–296 (2002)
8. Fang, K.-T., Maringer, D., Tang, Y., Winker, P.: Lower bounds and stochastic optimization algorithms for uniform designs with three or four levels. Math. Comput. **75**(254), 859–878 (2005)
9. Fang, K.-T., Tang, Y., Yin, J.: Lower bounds for wrap-around L_2-discrepancy and constructions of symmetrical uniform designs. Forthcomming (2004)
10. Fang, K.-T., Wang, Y.: Applications of Number Theoretic Methods in Statistics. Chapman and Hall, London (1994)
11. Lenth, R.V.: Quick and easy analysis of unreplicated factorials. Technometrics **31**, 469–473 (1989)
12. Lin, D.K.J., Sharpe, C., Winker, P.: Optimized U-type designs on flexible regions. Comput. Stat. Data Anal. **54**, 1505–1515 (2010)
13. Lin, D.K.J., Peng, J.Y.: The order-of-addition experiments: a review and some new thoughts (with discussion). Qual. Eng. **31**(1), 49–59 (2019)

14. Liu, M.Q., Hickernell, F.J.: $E(s^2)$-optimality and minimum discrepancy in 2-level superdaturated designs. Statistica Sinica **12**, 931–939 (2002)
15. Mee, R.W.: Order of addition modeling. Statistica Sinica. In press (2019)
16. Pukelsheim, F.: Optimal Design of Experiments. Wiley, New York (1993)
17. Peng, J.Y., Mukerjee, R., Lin, D.K.J.: Design of order-of-addition experiments. Biometrika. In press (2019)
18. Van Nostrand, R.C.: Design of experiments where the order of addition is important. In: ASA Proceedings of the Section on Physical and Engineering Sciences, pp. 155–160. American Statistical Association, Alexandria, Virginia (1995)
19. Volkel, J.G.: The design of order-of-addition experiments. J. Qual. Technol. (2019) https://doi.org/10.1080/00224065.2019.1569958
20. Winker, P.: Optimization Heuristics in Econometrics. Wiley, Chichester (2001)
21. Winker, P., Fang, K.-T.: Application of Threshold Accepting to the evaluation of the discrepancy of a set of points. SIAM J. Numer. Anal. **34**, 2028–2042 (1997)
22. Winker, P., Fang, K.-T.: Optimal U-type designs. In: Niederreiter, H., Hellekalek, P., Larcher, G., Zinterhof, P. (eds.) Monte Carlo and Quasi-Monte Carlo Methods, pp. 436–488. Springer, New York (1997)
23. Winker, P., Gilli, M.: Applications of optimization heuristics to estimation and modelling problems. Comput. Stat. Data Anal. **47**, 211–223 (2004)
24. Yang, J.F., Sun, F.S., Xu, H.: Component orthogonal arrays for order-of-addition experiments. Submitt. Manuscr. (2019)
25. Zhao, Y.N., Lin, D.K.J., Liu, M.Q.: Minimal-point design for order of addition experiment. Submitt. Manuscr. (2019)