Journal of Complexity 28 (2012) 224–237

Contents lists available at SciVerse ScienceDirect

Journal of Complexity

journal homepage: www.elsevier.com/locate/jco

Constructing uniform designs: A heuristic integer programming method

Yong-Dao Zhou [a,*], Kai-Tai Fang [b,c], Jian-Hui Ning [d]

[a] College of Mathematics, Sichuan University, Chengdu 610064, China
[b] BNU-HKBU United International College, Zhuhai, 519085, China
[c] Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing 100190, China
[d] College of Mathematics and Statistics, Central China Normal University, Wuhan, 430079, China

ARTICLE INFO

Article history:
Received 9 November 2010
Accepted 29 September 2011
Available online 14 October 2011

Keywords:
Quadratic integer programming
Simulated annealing
Uniform design
Wrap-around L_2-discrepancy

ABSTRACT

In this paper, the wrap-around L_2-discrepancy (WD) of asymmetrical design is represented as a quadratic form, thus the problem of constructing a uniform design becomes a quadratic integer programming problem. By the theory of optimization, some theoretic properties are obtained. Algorithms for constructing uniform designs are then studied. When the number of runs n is smaller than the number of all level-combinations m, the construction problem can be transferred to a zero–one quadratic integer programming problem, and an efficient algorithm based on the simulated annealing is proposed. When $n \geq m$, another algorithm is proposed. Empirical study shows that when n is large, the proposed algorithms can generate designs with lower WD compared to many existing methods. Moreover, these algorithms are suitable for constructing both symmetrical and asymmetrical designs.

1. Introduction

The main idea of the uniform design (UD) is to put experimental points uniformly scattered on the experimental domain [8], and it is one kind of computer experimental designs and one kind of physical experimental designs with model uncertainty [5]. The UD has gained prominence in recent years. Many authors have proposed a number of construction methods for UD such as the number theoretic method which includes the good lattice point (glp) method and the glp method with power generator (pglp method, see [8]), stochastic optimization algorithm [7,20], the combinatorial

* Corresponding author.
E-mail addresses: ydzhou@scu.edu.cn (Y.-D. Zhou), ktfang@uic.edu.hk (K.-T. Fang), jhning@mail.ccnu.edu.cn (J.-H. Ning).

0885-064X/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jco.2011.10.005

construction method [5], and so on. In practical applications, however, uniform design tables with large size (a large number of runs or/and a large number of factors) are urgent. It is known that generating a uniform design is an optimization problem, where the objective function is determined by a pre-given uniformity measure. In this paper, this optimization problem will be expressed as a quadratic integer programming problem. By using the theory of quadratic integer programming, some new construction methods for uniform designs are proposed.

As a measure of uniformity, the star L_p-discrepancy has been widely used in number-theoretic methods [8,18]. Hickernell [10,11] pointed out some weakness of the star L_p-discrepancy and proposed several modifications, among which the wrap-around L_2-discrepancy (WD) has nice properties. Fang and Ma [6] showed that the formula of WD of a symmetrical design can be reformulated as a quadratic form. In this paper, their result is extended to asymmetrical designs. Note that the theory of optimization including convex optimization and quadratic integer programming has been rapidly developed in the past decades (see [3,4,12]); some efficient algorithms for quadratic integer programming can be used to construct uniform designs for the two cases: (a) the number of runs n is less than the total number of level-combinations $m = q_1 \cdots q_s$ and (b) $n \geq m$.

For the first case, the uniform design may require that the design points do not overlap each other since the repeated points do not carry more information in computer experiments, and thus the corresponding quadratic programming problem may be transferred to a zero–one quadratic programming problem with some constraints. Furthermore, this constrained problem can be reformulated as an unconstrained zero–one quadratic programming problem (see Section 3). For solving the unconstrained zero–one quadratic programming problem in case (a), many methods were proposed in the literature, such as some trajectory methods including Tabu search [2,19] and simulated annealing (SA, see [2,14]), some population based methods including scatter search [1] and evolutionary algorithms [15, 17], and other local search heuristics [16]. More details about these heuristics can be read from [9]. Katayama and Narihisa [14] presented an SA-based heuristic to test on publicly available benchmark instances of size ranging from 500 to 2500 variables and compared them with other heuristics. Computational results indicate that this SA leads to high-quality solutions with a short CPU times. For solving this problem Merz and Freisleben [16] proposed a greedy heuristic and two local search algorithms: 1-opt local search and k-opt local search. Based on the above development, in this paper, an algorithm called the *SA-based integer programming method* (SA-IPM) is proposed to solve the special zero–one quadratic programming problem for constructing uniform designs. SA-IPM combines SA-based heuristic and 1-opt local search with the best improvement move strategy.

For the second case, the corresponding optimization problem can also be reformulated as a quadratic integer problem. But the variables are no longer binary since '$n > m$' means that some design points have to overlap each other. Thus, this is a more complex optimization problem than the zero–one quadratic programming problem. To reduce the computational complexity, we propose another new algorithm which bases on some so called *combination method* (CM). The main idea is to divide the problem into two steps, i.e., first construct a small design with the number of runs n_0 ($n_0 \equiv n \pmod m$), and then add the full factorial designs to get the final design. Some empirical study shows that this algorithm is efficient.

The remainder of this paper is organized as follows. Section 2 presents a quadratic form of the WD of an asymmetrical design and some properties of this form. Section 3 provides an algorithm, SA-based heuristic with 1-opt local search, to construct uniform designs with the number of runs $n < m$. Another algorithm for the case of $n > m$ is considered in Section 4. Finally, Section 5 gives some discussion and conclusions.

2. Quadratic form of WD

An asymmetrical design $U(n; q_1, \ldots, q_s)$ with n runs and s factors each having respectively $q_1, \ldots, q_s$ levels is called a U-type design if all the levels of each factor appear equally often. Most existing UDs are generated based on U-type designs. When all q_i's are equal to q, the design is called *symmetrical* and denoted by $U(n; q^s)$. Let $\mathcal{U}(n; q_1, \ldots, q_s)$ and $\mathcal{U}(n; q^s)$ be the sets of all $U(n; q_1, \ldots, q_s)$ and $U(n; q^s)$, respectively. By mapping $f : l \rightarrow (2l-1)/(2q_i)$, $l = 1, \ldots, q_i$, $i = 1, \ldots, s$, the n runs of $U(n; q_1, \ldots, q_s)$ are transformed into n points in $C^s = [0, 1]^s$. In this paper, we only focus on designs in the C^s. Let $\boldsymbol{X} = \{\boldsymbol{x}_i = (x_{i1}, \dots, x_{is}),\ i = 1, \dots, n\}$ be a set of n points in C^s. An analytical expression of squared WD($\boldsymbol{X}$) is given by (see [11])

$$\mathrm{WD}^2(\boldsymbol{X}) = -\left(\frac{4}{3}\right)^s + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{s}\left[\frac{3}{2} - |x_{ik} - x_{jk}| + |x_{ik} - x_{jk}|^2\right]. \tag{1}$$

Under WD a uniform design $U_n(q_1, \dots, q_s)$ has the minimum WD over $\mathcal{U}(n; q_1, \dots, q_s)$.

For a given $\boldsymbol{X} \in \mathcal{U}(n; q_1, \dots, q_s)$ let $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{X})$ be a column vector of $n(i_1, \dots, i_s)$ arranged lexicographically, where $n(i_1, \dots, i_s)$ is the number of runs at the level-combination $(i_1, \dots, i_s)$ in the design $\boldsymbol{X}$. Clearly, the length of $\boldsymbol{y}$ is $m = q_1 \cdots q_s$. It is obviously that any design can be decided by the corresponding $\boldsymbol{y}$. The vector $\boldsymbol{y}$ may be called as the *frequency vector*. When the elements of the vector $\boldsymbol{y}$ are all nonnegative integers, the corresponding design is called an *exact design*, otherwise called *continuous design*. Each U-type design is an exact design.

Lemma 1. *Let design $\boldsymbol{X} \in \mathcal{U}(n; q_1, \dots, q_s)$ and $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{X})$, then*

$$\mathrm{WD}^2(\boldsymbol{X}) = -\left(\frac{4}{3}\right)^s + \frac{1}{n^2}\boldsymbol{y}'\boldsymbol{A}\boldsymbol{y}, \tag{2}$$

where $\boldsymbol{A} = A_1 \otimes A_2 \otimes \cdots \otimes A_s$,

$$A_k = (t_{ij}^k), \quad t_{ij}^k = \frac{3}{2} - \frac{|i-j|(q_k - |i-j|)}{q_k^2}, \ i, j = 1, \dots, q_k,\ k = 1, \dots, s, \tag{3}$$

and $\otimes$ denotes the Kronecker product.

Lemma 1 is an extension of Theorem 4.1 in [6] and we omit the proof. It can be checked that A_k $(k = 1, \dots, s)$ in (3) are positive semidefinite matrices, so does $\boldsymbol{A}$. Moreover, they also have some nice properties such as

$$\begin{cases} A_k\mathbf{1}_{q_k} = \left(\dfrac{4q_k}{3} + \dfrac{1}{6q_k}\right)\mathbf{1}_{q_k}, & \boldsymbol{A}\mathbf{1}_m = \displaystyle\prod_{k=1}^{s}\left(\frac{4q_k}{3} + \frac{1}{6q_k}\right)\mathbf{1}_m, \\ A_k^{-1}\mathbf{1}_{q_k} = \left(\dfrac{4q_k}{3} + \dfrac{1}{6q_k}\right)^{-1}\mathbf{1}_{q_k}, & \boldsymbol{A}^{-1}\mathbf{1}_m = \displaystyle\prod_{k=1}^{s}\left(\frac{4q_k}{3} + \frac{1}{6q_k}\right)^{-1}\mathbf{1}_m, \end{cases} \tag{4}$$

where $\mathbf{1}_t$ is the t-column vector of one's and $m = q_1 \cdots q_s$. For any exact design $U(n; q_1, \dots, q_s)$, every element of $\boldsymbol{y} = (y_1, \dots, y_m)'$ should be a non-negative integer. Now for given $q_1, \dots, q_s$ and n, based on the formula (2), the problem of constructing a uniform design $U_n(q_1, \dots, q_s)$ can be formulated as the following optimization problem:

$$(\mathrm{OP}) \quad \begin{cases} \min & f_0(\boldsymbol{y}) = -\left(\dfrac{4}{3}\right)^s + \dfrac{1}{n^2}\boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} \\ \text{s.t.} & \mathbf{1}_m'\boldsymbol{y} = n, \quad \boldsymbol{y} \in Z_+^m, \end{cases}$$

where $Z_+^m = Z_+ \times \cdots \times Z_+, Z_+ = \{0, 1, 2, \dots\}$, $m = q_1 \cdots q_s$, and $\boldsymbol{y} = (y_1, \dots, y_m)' \in Z_+^m$ means $y_i \in Z_+$. Actually, from the constraint $\mathbf{1}_m'\boldsymbol{y} = n$ and $\boldsymbol{y} \in Z_+^m$, we can reduce the range Z_+^m to Z_n^m, where $Z_n = \{0, 1, 2, \dots, n\}$. Thus, the problem (OP) is equivalent to the following optimization problem:

$$(\mathrm{OP}') \quad \begin{cases} \min & f_1(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} \\ \text{s.t.} & \mathbf{1}_m'\boldsymbol{y} = n, \quad \boldsymbol{y} \in Z_n^m. \end{cases}$$

By the equivalence between problem (OP) and problem (OP′), it is meant that they have the same solution. Note that problem (OP′) is not a convex optimization problem [3]. However, if the constraint $\boldsymbol{y} \in Z_n^m$ is relaxed, we have the following convex optimization problem

$$(\mathrm{SDP}) \quad \begin{cases} \min & f_2(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} \\ \text{s.t.} & \mathbf{1}_m'\boldsymbol{y} = n. \end{cases}$$

Problem (SDP) is a special convex quadratic programming, i.e., a semidefinite programming. Therefore, we can use the theory of convex optimization to solve problem (SDP). It is known that the Lagrange dual function is an effective tool for solving the problem (SDP) (see [3]). We have the following result.

Theorem 1. *The minimizer of problem (SDP) is*

$$\boldsymbol{y}^* = \frac{n}{m}\mathbf{1}_m, \tag{5}$$

where $m = q_1 \cdots q_s$.

Proof. It is well known that the Lagrangian associated with the problem (SDP) is

$$L(\boldsymbol{y}, \mu) = \boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} + \mu(\mathbf{1}'\boldsymbol{y} - n) = \boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} + \mu\mathbf{1}'\boldsymbol{y} - \mu n, \tag{6}$$

where $\mu \in R$ is the Lagrange multiplier associated with the constraint $\mathbf{1}'\boldsymbol{y} = n$. The Lagrange dual function can be obtained as follows:

$$g(\mu) = \inf_{\boldsymbol{y}} L(\boldsymbol{y}, \mu) = \inf_{\boldsymbol{y}} \left(\boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} + \mu\mathbf{1}'\boldsymbol{y}\right) - \mu n. \tag{7}$$

Derive the Eq. (6) with respect to $\boldsymbol{y}$ and let the derivative be zero, and we have

$$\hat{\boldsymbol{y}} = -\frac{1}{2}\mu\boldsymbol{A}^{-1}\mathbf{1}. \tag{8}$$

Thus

$$g(\mu) = -\frac{1}{4}\mu^2\mathbf{1}'\boldsymbol{A}^{-1}\mathbf{1} - \mu n, \tag{9}$$

which gives a lower bound of the optimal value $\boldsymbol{y}^*$ of the optimization problem (SDP). And the Lagrange dual problem associated with the problem (SDP) becomes

$$\max g(\mu). \tag{10}$$

Denote d^* and p^* are respectively the optimal value of problem (SDP) and the Lagrange dual problem (10). Since the problem (SDP) is a semidefinite programming, the optimal duality gap is zero, which means $d^* = p^*$ (see [3]). From (9), the maximizer of problem (10) is

$$\mu^* = -\frac{2n}{\mathbf{1}'\boldsymbol{A}^{-1}\mathbf{1}}. \tag{11}$$

Substituting (11) into (8), we have

$$\hat{\boldsymbol{y}} = -\frac{1}{2}(\boldsymbol{A}^{-1}\mathbf{1})\left(-\frac{2n}{\mathbf{1}'\boldsymbol{A}^{-1}\mathbf{1}}\right). \tag{12}$$

Finally by the properties of $\boldsymbol{A}$ in (4), we obtain the minimizer $\boldsymbol{y}^* = \frac{n}{m}\mathbf{1}$, which completes the proof. □

Remark 1. Theorem 1 is an extension of Theorem 4.2 in [6], but the current proof is simpler.

Remark 2. According to Theorem 1, usually the optimal design $\boldsymbol{X}$ associated with $\boldsymbol{y}^*$ is not an exact design. However, a full design with $n = km$ and $\boldsymbol{y} = k\mathbf{1}$ for some positive integer k is a uniform design under WD.

Corollary 1. *The squared* WD *of a full design $\boldsymbol{X} \in \mathcal{U}(n; q_1, \ldots, q_s)$ with $n = km$ runs is given by*

$$\mathrm{WD}^2(\boldsymbol{X}) = \prod_{i=1}^{s}\left(\frac{4}{3} + \frac{1}{6q_i^2}\right) - \left(\frac{4}{3}\right)^s, \tag{13}$$

where $m = q_1 \cdots q_s$.

It can be seen that the squared WD-value of a full design is independent of k. If we substitute (5) into (2) a lower bound of WD^2 is given by the right hand side of (13). So a full design is a uniform design.

Remark 3. When $n \neq km$, k is a positive integer, it is easily known that the optimal value $\boldsymbol{y}^*$ of problem (SDP) is not the minimizer of problem (OP), and there exists a duality gap between problem (SDP) and problem (OP) depending on k.

3. Algorithm for the case of $n < m$

In this section, we propose an algorithm for constructing uniform designs in the case that the number of runs n is less than $m = q_1 \cdots q_s$. It is reasonable to require that in this case the resulted design should have no coincident points. Let the frequency vector of the design $U(n; q_1, \ldots, q_s)$ be $\boldsymbol{y} = (y_1, \ldots, y_m)'$, where $y_i \in \{0, 1\}$ and $\sum_{i=1}^m y_i = n$. And every element of $\boldsymbol{y}$ has the property $y_i^2 = y_i$.

It is well known that quadratic integer programming problem (OP$'$) with a linear constraint is equivalent to the following unconstrained optimization problem (see [13]):

$$(\mathrm{OP}'') \quad \begin{cases} \min & f_3(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{A}\boldsymbol{y} + K(\boldsymbol{A})(\boldsymbol{1}_m'\boldsymbol{y} - n)^2 \\ \text{s.t.} & \boldsymbol{y} \in \{0, 1\}^m, \end{cases}$$

where $\boldsymbol{y} \in \{0, 1\}^m$ means that each $y_i \in \{0, 1\}$, and $K(\boldsymbol{A}) = 2\sum_{i=1}^m \sum_{j=1}^m |a_{ij}| + 1$ where $A = (a_{ij})$ is defined in Lemma 1. Furthermore, because of the property $y_i^2 = y_i$, we can rewrite the problem (OP$''$) as follows:

$$(\mathrm{OP}''') \quad \begin{cases} \min & f_4(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{Q}_0\boldsymbol{y} + K(\boldsymbol{A})n^2 \\ \text{s.t.} & \boldsymbol{y} \in \{0, 1\}^m, \end{cases}$$

where $\boldsymbol{Q}_0 = \boldsymbol{A} + K(\boldsymbol{A})\boldsymbol{1}_m\boldsymbol{1}_m' - 2K(\boldsymbol{A})m\boldsymbol{I}_m$ is a symmetric matrix. Note that the value of $K(\boldsymbol{A})$ is large in most cases and the elements of $\boldsymbol{Q}_0$ is also large; it is better to divide the objective function by $K(\boldsymbol{A})$ and to remove the constant $K(\boldsymbol{A})n^2$, and problem (OP) in the case of $n < m$ can be rewritten as the following unconstrained quadratic problem:

$$(\mathrm{BQP}) \quad \begin{cases} \min & f(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{Q}_{\boldsymbol{y}} \\ \text{s.t.} & \boldsymbol{y} \in \{0, 1\}^m, \end{cases}$$

where $\boldsymbol{Q} = \boldsymbol{Q}_0/K(\boldsymbol{A})$. In the literature, the problem (BQP) belongs to the *unconstrained binary quadratic programming problem*. This problem is also known as the unconstrained quadratic bivalent programming problem or the unconstrained quadratic zero–one programming problem (see [2]).

3.1. Simulated annealing-based heuristic

To deal with the computational complexity of the problem (BQP) for large m, we combine the SA algorithm and local search algorithm to generate uniform designs and call this algorithm as the SA-based integer programming method.

Algorithm 1 The SA-IPM for constructing uniform designs in the case of $n < m$

```
1:  Initialize $I, J, T_{init}, T_f, T_r$;
2:  Generate an initial random solution $\boldsymbol{y}_0 \in \{0, 1\}^m$;
3:  for $i = 1 : I$
4:     Set $\boldsymbol{y} = \boldsymbol{y}_0, T = T_{init}$, ct $= 0$;
5:     Calculate gains $g_i$ of $\boldsymbol{y}$ for all $i$ in $\{1, \ldots, m\}$;
6:     while ct $< J$;
```

```
 7:       Set ct $=$ ct $+ 1$;
 8:       for $t = 1 : m$
 9:          Find $j$ with $g_j = \min_k g_k$;
10:          If $g_j < 0$, then set ct $= 0$ and $y_j = 1 - y_j$ (and update all gains $g_i$);
11:          Otherwise, random choose $k \in \{1, \ldots, m\}$, set $y_k = 1 - y_k$ with probability
        $e^{-g_j/T}$ (and update all gains $g_i$);
12:       end
13:       Set $T = T_f \times T$;
14:    end
15:    If the design with respect to $\boldsymbol{y}$ reaches its lower bound, return $\boldsymbol{y}$;
16:    Otherwise, set $\boldsymbol{y}_0 = \boldsymbol{y}$, $T_{init} = T_r \times T_{init}$;
17: end
18: Return $\boldsymbol{y}$;
```

The pseudo-code of SA-IPM can be seen in Algorithm 1. Parameters I, J, T_{init} represent the number of time of annealing process, the termination conditional number at each iteration, the initial temperature, respectively. Parameters $T_f, T_r \in (0, 1)$ are two temperature reduction rates. Define a neighbor of current solution $\boldsymbol{y} = (y_1, \ldots, y_m)$ as

$$\{\boldsymbol{y}_i = (y_1, \ldots, y_{i-1}, 1 - y_i, y_{i+1}, \ldots, y_m),\ i = 1, \ldots, m\},$$

so the hamming distance between $\boldsymbol{y}_i$ and $\boldsymbol{y}$ is equal to 1. Define the gain $g_i = f(\boldsymbol{y}_i) - f(\boldsymbol{y})$, where $f(\cdot)$ is the objective function in problem (BQP), and $g_i < 0$ means $\boldsymbol{y}_i$ is a good neighbor, otherwise it is a bad one. According to [16], the gain g_i can be calculated by

$$g_i = q_{ii}(\bar{y}_i - y_i) + 2 \sum_{k=1, k \neq i}^{m} q_{ki} y_k (\bar{y}_i - y_i), \tag{14}$$

where $\bar{y}_i = 1 - y_i$, q_{ki} is the (k, i)-element of the matrix $\boldsymbol{Q}$ in problem (BQP). The gain g_i can be calculated in a linear time of m, but all gains of neighbors must be calculated in $O(m^2)$ time. However, the gains g_i do not have to be recalculated each time. Assuming that all g_i for a current solution have been calculated and the bit k is flipped, we can compute the new gain g_i^n efficiently with the formula:

$$g_i^n = \begin{cases} -g_i, & \text{if } i = k, \\ g_i + 2q_{ik}(\bar{y}_i - y_i)(\bar{y}_k - y_k), & \text{otherwise}, \end{cases} \tag{15}$$

and the update gains can be performed in a linear time. Step (9:) includes a local search for the best improvement, which is different from the classical SA. It is possible in Step (10:) that there are different j satisfied $g_j = \min_i g_i$. In this case, we randomly choose one bit to flip. In Steps (10:) and (11:), all gains g_i can be updated by using (15). Moreover, if the lower bound in Step (10:) is reached, the process is terminated. The lower bound of design under WD can be seen in [7,21].

3.2. Computational results

This subsection shows performance of the SA-IPM for construction of uniform designs under WD. For comparisons among the SA-IPM, the glp method (see [8]) and the existing designs on the web (http://www.math.hkbu.edu.hk/UniformDesign/) we restrict our study only on construction of symmetric uniform designs. All the results below are obtained by using Matlab in a personal computer with 2.1 GHz CPU processor.

Initial values of the parameters in SA-IPM are set by preliminary testing, i.e., for different cases in Table 1, a unified setting of the parameters is considered to have a trade-off between the quality of the resulted design and computer running time. In our simulation, we set $T_{init} = 1/q$, where q is the number of levels, $J = 10$ and the ratio $T_f = 0.99$. For the parameters I and T_r, from our experience, we consider $I = 10, T_r = 0.9$ when $m < 500$, and $I = 2, T_r = 0.8$ when $m \geq 500$. Since the initial solution in SA-IPM is generated randomly, we repeat $K = 30$ times of SA-IPM for constructing uniform design and choose the best one as the final solution.

Table 1
Comparisons between SA-IPM and other constructing methods.

Case no.	n	s	q	m	SA-IPM	Time (s)	C_5	C_{95}	RGM	Designs on web	glp or pglp method	Time (s)
1	9	3	3	27	0.100956 (−2.3074)	1.3	0	0	0.3182	0	0.3170	0.1
2	15	3	3	27	0.101118 (−2.1946)	1.3	0	0	0.2673	0	0.7961	0.1
3	48	5	3	243	0.302871 (−2.7553)	12.3	0.0056	0.0271	0.7833	−0.0298	0.0582	116.2
4	102	5	3	243	0.301352 (−2.7897)	12.1	0.0057	0.0237	0.8170	–	0.0532	∗
5	201	5	3	243	0.300992 (−2.7549)	12.0	0.0061	0.0326	0.7833	–	0.4063	∗
6	48	7	3	2187	0.774745 (−3.4149)	1015.4	0.0274	0.0902	1.2904	−0.1291	0.0033	∗
7	201	7	3	2187	0.760866 (−3.5557)	1010.5	0.0019	0.0243	1.4561	–	0.2444	∗
8	1002	7	3	2187	0.759480 (−3.6064)	1024.4	0.0042	0.0140	1.4977	–	0.1138	∗
9	36	3	4	64	0.056460 (−2.3990)	4.3	0.0012	0.0184	0.4757	−0.0107	0.0456	0.4
10	36	4	4	256	0.101575 (−2.7944)	55.4	0.0000	0.0509	0.7775	−0.0146	0.0789	3.4
11	100	4	4	256	0.100141 (−2.8317)	55.3	0.0001	0.0420	0.8194	–	0.0702	395.7
12	200	4	4	256	0.099960 (−2.8544)	55.4	0.0383	0.0793	0.8479	–	0.1901	2776.5
13	36	5	4	1024	0.172340 (−3.1839)	147.1	0.0259	0.0951	1.0993	−0.0384	0.1008	31.6
14	200	5	4	1024	0.167471 (−3.2793)	150.7	0.0028	0.0330	1.1962	–	0.0843	∗
15	500	5	4	1024	0.167245 (−3.3043)	152.1	0.0180	0.0300	1.2017	–	0.1120	∗
16	55	3	5	125	0.035994 (−2.5761)	30.8	0.0054	0.0218	0.6494	−0.0146	0.0345	0.3
17	100	4	5	625	0.064121 (−3.0392)	86.9	0.0024	0.0267	1.0084	–	0.0389	395.0
18	500	4	5	625	0.063705 (−3.0476)	89.3	0.0010	0.0162	1.0160	–	0.1149	∗
19	200	5	5	3125	0.106927 (−3.5258)	1336.4	0.0026	0.0226	1.4146	–	0.0147	∗
20	1000	5	5	3125	0.106440 (−3.6025)	1326.6	0.0006	0.0067	1.4828	–	0.0065	∗
21	60	3	6	216	0.025030 (−2.7025)	68.5	0.0062	0.0421	0.7520	−0.0233	0.0362	1.5
22	204	4	6	1296	0.044286 (−3.2200)	334.0	0.0062	0.0205	1.1598	–	0.0270	∗
23	504	4	6	1296	0.044158 (−3.2295)	333.0	0.0043	0.0121	1.1942	–	0.0188	∗

The notation "–" in 8th column means the corresponding design does not exist on web yet.
The notation "∗" in last column means the design is constructed by the pglp method.

For comparing performance of Algorithm 1 with other constructing methods, such as the glp method or its modifications [8] and Threshold-Accepting algorithm [7,20], as a benchmark we also consider the mean and standard deviation of WDs of the designs constructed by the random generating method (RGM). The procedure of RGM is as follows: first randomly generate zero–one vector $\boldsymbol{y} \in \{0, 1\}^m$, where the number of element 1 in the vector is equal to n, and this $\boldsymbol{y}$ corresponds to a design with the number of runs n according to the lexicographical order of the m runs. Secondly we calculate WD of this design. Repeat this procedure N times; we obtain N WD values, their mean (M_{rgm}), standard deviation (std_{rgm}), and 0.1% percentile of the WD values, which is denoted as WD value by RGM (WD_{rgm}). In this work we take $N = 10^7$.

Table 1 shows comparisons among the three methods: SA-IPM, glp and RGM in terms of the computational time and WD-values for different n, s, q and m. Denote by WD_{sa}, WD_{glp}, WD_{rgm} and WD_0 the WD of design obtained by SA-IPM, glp, RGM and the WD of existing design on web, respectively. Column 6 stands for WD-value of WD_{sa} and $\text{WD}_{\text{stand}} = (\text{WD}_{\text{sa}} - M_{\text{rgm}})/\text{std}_{\text{rgm}}$. Column 7 denotes the running time of SA-IPM with 30 repeated times. Moreover, since the procedure of SA-IPM is repeated several times, the percentiles of their WD-values are considered to compare with many existing methods. Let WD_p be the pth percentile and $C_p = (\text{WD}_p - \text{WD}_{\text{sa}})/\text{std}_{\text{rgm}}$. Then, columns 8 and 9 show C_5 and C_{95}, respectively. And columns 10–12 give respective values of $L_{\text{rgm}} = (\text{WD}_{\text{rgm}} - \text{WD}_{\text{sa}})/\text{std}_{\text{rgm}}$, $L_0 = (\text{WD}_0 - \text{WD}_{\text{sa}})/\text{std}_{\text{rgm}}$ and $L_{\text{glp}} = (\text{WD}_{\text{glp}} - \text{WD}_{\text{sa}})/\text{std}_{\text{rgm}}$. The last column denotes the running time of the glp method.

From Table 1, several conclusions can be drawn as follows:

(1) When $m = 27$, the SA-IPM can deliver a design with lowest WD as well as the existing design on web, where $C_{95} = 0$ means that almost every procedure of SA-IPM can obtain a design with lowest WD.
(2) Uniformity of the design constructed by SA-IPM is, in general, better than the corresponding design constructed by the RGM and glp method, since all the values of L_{glp} and L_{rgm} are positive.
(3) C_5-value is smaller than L_{glp} in each case except for case 6, and C_{95}-value is smaller than L_{glp} except for cases 6, 19, 20, 21. Moreover, the L_{rgm}-value is larger than C_5, C_{95} and L_{glp} in every case.

Table 2
The number of iterations and running time of SA-IPM when the resulted designs come as close a 2% to the best known solution.

Case	1	2	3	6	9	10	13	16	21
Iterations	19634.1	1902.4	12 320.1	95 681.3	5146.2	26 170.9	908 134.4	15 102.5	35 398.1
Close rate	0.0000	0.0046	0.0070	0.0152	0.0080	0.0127	0.0168	0.0088	0.0134
Running time	0.2155	0.0213	0.2145	7.9584	0.0622	0.4667	39.4091	0.2102	0.5947

Therefore, the RGM is the worst one among the four methods and the SA-IPM has robustness in a certain sense.

(4) Except for the first two cases, the uniformity of the delivered designs obtained by SA-IPM is worse than that of the corresponding existing designs on web. A possible reason is that the number of iterations in SA-IPM is not enough. In many practical situations, the user often urgently needs a design that can meet the request. The running time for obtaining a required design is an important issue. The uniform designs under WD on the web are usually constructed by the Threshold-Accepting (TA) algorithm (see [7,20]). Due to the computational complexity the number of runs in these designs is limited to 60. When $n > 200$, the TA algorithm is almost unaffordable. If SA-IPM can deliver a design that is very close to the 'best' one in the sense of WD with a significant shorter time, it will be much helpful in practice. How to measure close to the 'best'? Let WD_0 be the WD of the 'best' design and WD be the WD of a design generated by a fact algorithm. The ratio $\mathrm{WD}_{\mathrm{ratio}} = (\mathrm{WD} - \mathrm{WD}_0)/\mathrm{WD}_0$ can be used to measure the closeness. Most designs in Table 1 have $\mathrm{WD}_{\mathrm{ratio}} < 0.04$. For example, for case 6 $\mathrm{WD}_0 = 0.768411$ and its $\mathrm{WD}_{\mathrm{ratio}} = (0.774745 - 0.768411)/0.768411 = 0.0082$.

For a further study, consider cases 1, 2, 3, 6, 9, 10, 13, 16 and 21 and fix the values of parameters in SA-IPM as that in Table 1. We choose different number of iterations and the corresponding running time so that the delivered design generated by SA-IPM has $\mathrm{WD}_{\mathrm{ratio}} < 0.02$. Table 2 lists the number of iterations and the corresponding running time. Since SA-IPM is a stochastic algorithm, we repeat 100 times and calculate the mean of the number of iterations and running times for these cases. It is shown that the designs obtained by SA-IPM can quickly get close to the best known design.

(5) When fixed the parameters in the procedure of SA-IPM, its running time mainly depends on the parameter $m = q^s$ and does not much depend on the number of runs n. For example, in cases 13, 14, and 15, $m = 1024$ and n varies from 36 to 500, the running times of SA-IPM are all near 150 s. On the other hand, the running time of the glp method is much depend on n and is very long when $n \geq 200$ and $s \geq 5$. Therefore, when $m < 3200$ and $n \geq 200$, the SA-IPM is a better method to construct uniform designs.

Now we consider the effect of three parameters: the number of repeating times, K, and the number of iterations that are determined by I and J in SA-IPM. The larger the parameters I and J in SA-IPM are, the larger the number of iterations has. Cases 5, 18 and 22 in Table 1 are selected for testing.

Let us increase the number of repeating times K from 30 to 570 and keep other parameters unchanged. For each case and for a given number of repeating times K, the design with the lowest WD-value among K output designs is considered as the delivered design. The quantity $\mathrm{WD}_{\mathrm{stand}} = (WD_{\mathrm{sa}} - M_{\mathrm{rgm}})/\mathrm{std}_{\mathrm{rgm}}$ is used for comparisons.

For the three cases 5, 18 and 22 Fig. 1(a) shows $\mathrm{WD}_{\mathrm{stand}}$ when the number of repeating times is increased. Fig. 1(b) shows $\mathrm{WD}_{\mathrm{stand}}$ under different values of J with $I = 2$, and Fig. 1(c) shows $\mathrm{WD}_{\mathrm{stand}}$ under different values of I with $J = 10$. Fig. 1(b) and (c) show that the increase of the value of J and I do not improve the result, while the number of iterations increases very much. Therefore, the choice of the values of parameters in Table 1 is reasonable and larger number of iterations does not necessarily to obtain better designs.

Last, an interesting result of these cases in Table 1 is that all of these designs constructed by SA-IPM are U-type designs.

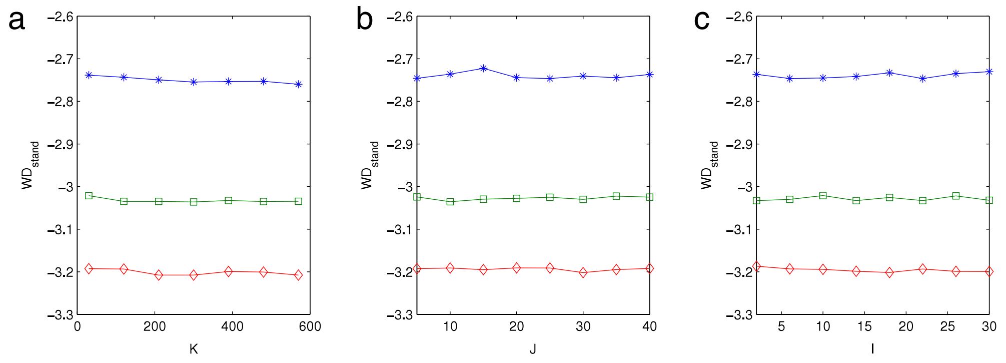

Fig. 1. WD_{stand} of the delivered design obtained by SA-IPM, when (a) the repeated times K is increased from 30 to 570, (b) the parameter J is increased from 5 to 40, and (c) the parameter I is increased from 2 to 30. "–∗", "–□", "–◊" stand for case 5, 18 and 22, respectively.

3.3. *The enumeration method*

Sometimes the number of runs $n < m = q_1 \cdots q_s$ and $\binom{m}{n}$ is not too large. In this case the $N_n = \binom{m}{n}$ frequency vectors $\boldsymbol{y} = (y_1, \ldots, y_m)'$ can be enumerated. We can find a design with the minimum WD-value, denoted by D_0, among these $\binom{m}{n}$ designs. Obviously, D_0 may be or may not be a U-type design. On the other hand, it is known that the number of U-type designs in $\mathcal{U}(n; q_1, \ldots, q_s)$ is $N_u = \prod_{i=1}^{s}\left[\binom{n}{n/q_i}\binom{n-n/q_i}{n/q_i}\cdots\binom{2n/q_i}{n/q_i}\right]$. Therefore, the cardinality of the candidate sets by using the enumeration method may be much smaller than N_u. For example, for constructing a uniform design $U_{60}(4^3)$, the enumeration method needs to compare $N_n = \binom{64}{60} = 635\,376$ designs, while the number of all the U-type designs is $N_u \approx 2.3042 \times 10^{100}$, which is too large for using some stochastic optimization algorithm directly. In this case, the enumeration method is a better choice.

Using the enumeration method, we can easily construct the uniform designs $U_3(3^2)$, $U_6(3^2)$, $U_4(4^2)$, $U_8(4^2)$, $U_{12}(4^2)$, $U_5(5^2)$, $U_{20}(5^2)$, $U_6(3^3)$, $U_{21}(3^3)$, $U_{24}(3^3)$, $U_4(4^3)$, $U_{60}(4^3)$ and $U_{78}(3^4)$, which have the same WD as the existing designs, and the computational speed of every case is very fast (several seconds). It is interesting that all of these resulted design are U-type designs, which means the U-type constraint is reasonable for constructing uniform designs by some stochastic optimization methods.

4. Algorithm for the case of $n > m$

It is known from Section 2 that a full factorial design is a uniform design when $n = tm$ and t is a positive integer. In this section, we consider some construction methods of the uniform design when the number of runs $n > m = q_1 \cdots q_s$ and m is not a divisor of n. Denote $n = n_0 + tm$, where $1 \le n_0 < m$ and $t \ge 1$.

4.1. *Combined method*

Let $\boldsymbol{y} = \boldsymbol{y}_1 + \boldsymbol{y}_2$, where $\mathbf{1}_m'\boldsymbol{y}_1 = n_0$, $\mathbf{1}_m'\boldsymbol{y}_2 = tm$. The objective function in problem (OP) can be rewritten as

$$\boldsymbol{y}'\mathbf{Q}\boldsymbol{y} = (\boldsymbol{y}_1 + \boldsymbol{y}_2)'\mathbf{Q}(\boldsymbol{y}_1 + \boldsymbol{y}_2) = \boldsymbol{y}_1'\mathbf{Q}\boldsymbol{y}_1 + 2\boldsymbol{y}_1'\mathbf{Q}\boldsymbol{y}_2 + \boldsymbol{y}_2'\mathbf{Q}\boldsymbol{y}_2, \tag{16}$$

with the constraint $\mathbf{1}_m'\boldsymbol{y} = n$, and the objective function (16) can be thought as a multivariate quadratic integer programming problem. Let $f_5(\boldsymbol{y}) = \boldsymbol{y}_1'\mathbf{Q}\boldsymbol{y}_1$ and $f_6(y) = 2\boldsymbol{y}_1^{*\prime}\mathbf{Q}\boldsymbol{y}_2 + \boldsymbol{y}_2'\mathbf{Q}\boldsymbol{y}_2$, where $\boldsymbol{y}_1^*$ is the optimizer of $f_5(\boldsymbol{y})$ with the constraint $\mathbf{1}_m'\boldsymbol{y}_1 = n_0$. Suppose the minimum of $f_5(\boldsymbol{y})$ and $f_6(\boldsymbol{y})$ under the constraint $\mathbf{1}_m'\boldsymbol{y}_1 = n_0$, $\mathbf{1}_m'\boldsymbol{y}_2 = tm$ are respectively M_5 and M_6, then we know that the minimum of $\boldsymbol{y}'\boldsymbol{Q}\boldsymbol{y}$ should be less than or equal to $M_5 + M_6$. However, it is difficult to directly solve the multivariate quadratic integer programming problem (16) due to its computational complexity. Then, the procedure of solving the optimization problem (OP) can be divided into the following two parts

$$\text{(OP1)} \quad \begin{cases} \min & f_5(\boldsymbol{y}) = \boldsymbol{y}_1'\boldsymbol{Q}\boldsymbol{y}_1 \\ \text{s.t.} & \boldsymbol{1}_m'\boldsymbol{y}_1 = n_0, \quad \boldsymbol{y}_1 \in Z_{n_0}^m, \end{cases}$$

and

$$\text{(OP2)} \quad \begin{cases} \min & f_6(y) = 2\boldsymbol{y}_1^*\boldsymbol{Q}\boldsymbol{y}_2 + \boldsymbol{y}_2'\boldsymbol{Q}\boldsymbol{y}_2 \\ \text{s.t.} & \boldsymbol{1}_m'\boldsymbol{y}_2 = tm, \quad \boldsymbol{y} \in Z_{tm}^m, \end{cases}$$

where $\boldsymbol{y}_1^*$ is the minimizer of problem (OP1). Here, $\boldsymbol{y}_1^*$ can be obtained by the SA-IPM proposed in the last section and it satisfies $\boldsymbol{1}_m'\boldsymbol{y}_1^* = n_0$. Assume $\boldsymbol{y}_2^*$ be the minimizer of problem (OP2), then the final solution is $\hat{\boldsymbol{y}} = \boldsymbol{y}_1^* + \boldsymbol{y}_2^*$. Although the final solution $\hat{\boldsymbol{y}}$ may not reach the real minimizer of problem (OP), but this partition can reduce the computational complex significantly. In next subsection, we will show that this two-step method for constructing uniform designs is better than many existing construction methods.

Now, we consider the minimizer of problem (OP2) based on the known $\boldsymbol{y}_1^*$. Similarly, problem (OP2) can be relaxed as following problem:

$$\text{(SDP2)} \quad \begin{cases} \min & f(\boldsymbol{y}) = \boldsymbol{y}_2'\boldsymbol{Q}\boldsymbol{y}_2 + 2\boldsymbol{y}_1^*\boldsymbol{Q}\boldsymbol{y}_2 \\ \text{s.t.} & \boldsymbol{1}_m'\boldsymbol{y}_2 = tm. \end{cases}$$

In fact, problem (SDP2) is a standard form of semidefinite program of the parameter $\boldsymbol{y}_2$ and we have the following theorem which provides the solution.

Theorem 2. *The minimizer of problem* (SDP2) *is*

$$\boldsymbol{y}_2^* = \left(\frac{n_0}{q^s}\boldsymbol{1}_m - \boldsymbol{y}_1^*\right) + t\boldsymbol{1}_m, \tag{17}$$

where $\boldsymbol{y}_1^*$ is the optimal vector in problem (OP1).

The proof of Theorem 2 is similar to that of Theorem 1, and we omit it. In problem (OP1), $\boldsymbol{y}_1^*$ is an integer-vector. If this constraint is relaxed to real-vector, then from the result in Theorems 1 and 2 indicates that $\boldsymbol{y}_2^* = t\boldsymbol{1}_m$. Therefore, it is reasonable to take $\boldsymbol{y}_2^* = t\boldsymbol{1}_m$.

Remark 4. It is possible that the WD under different frequency vectors $\boldsymbol{y}_1^1$ and $\boldsymbol{y}_1^2$ have the same value. For example, when $n = 6, q = 3, s = 2$, the designs with respect to $\boldsymbol{y}_1^1 = (1, 1, 0, 0, 1, 1, 1, 0, 1)'$ and $\boldsymbol{y}_1^2 = (1, 1, 0, 1, 0, 1, 0, 1, 1)'$ have the same $\text{WD}^2 = 0.0525$, and both designs are uniform design $U_6(3^2)$.

The following theorem gives some related property.

Theorem 3. *Let designs $D_1, D_2 \in \mathcal{U}(n_0; q_1, \ldots, q_s)$ have the same* WD^2 *values* WD_0 *and denote their frequency vectors by $\boldsymbol{y}_1^1$ and $\boldsymbol{y}_1^2$, respectively. Then the designs $U_1, U_2 \in \mathcal{U}(n; q_1, \ldots, q_s)$ whose frequency vectors are respectively $\boldsymbol{y}_1^1 + t\boldsymbol{1}_m$ and $\boldsymbol{y}_1^2 + t\boldsymbol{1}_m$, also have the same WD value,*

$$\begin{aligned} \text{WD}^2(U_1) &= \text{WD}^2(U_2) \\ &= \left(\frac{n_0^2}{n^2} - 1\right)\left(\frac{4}{3}\right)^s + \frac{1}{n^2}\left(n_0\text{WD}_0 + (2tn_0 + t^2m)\prod_{k=1}^s\left(\frac{4q_k}{3} + \frac{1}{6q_k}\right)\right) \end{aligned}$$

where t is a positive integer and $n = n_0 + tm$, $1 \le n_0 \le m$.

By Theorem 2 and the properties of $\boldsymbol{Q}$ in (4), the proof of Theorem 3 is straightforward. If $n_0 = m$, from Theorem 1, the optimal frequency vector $\boldsymbol{y}_1^1 = \boldsymbol{y}_1^2 = \boldsymbol{1}_m$ and the result in Theorem 3 is same as the result in Corollary 1, i.e., $\mathrm{WD}^2(U_1) = \mathrm{WD}_0$. Moreover, based on Theorems 2 and 3, we propose Algorithm 2 below, called the *combination method* (CM), to construct uniform designs with a large number of runs.

```
Algorithm 2 Combination method for constructing uniform design with $n = n_0 + tm$
(1) Use some algorithms to find the minimizer of the problem (OP1), denoted as $\boldsymbol{y}_1^*$;
(2) Choose $\boldsymbol{y}_2 = t\mathbf{1}_m$, where $m = q_1 \cdots q_s$;
(3) the frequency vector of final design is $\boldsymbol{y}^* = \boldsymbol{y}_1^* + t\mathbf{1}_m$.
```

From Remark 4, the optimal vector of $\boldsymbol{y}_1^*$ may not be unique and we can choose any one of $\boldsymbol{y}_1^*$ in Step (1) of Algorithm 2. Algorithm 2 shows that for constructing a uniform design $U_n(q_1, \ldots, q_s)$ when $n > m$, one first constructs the design $U_{n_0}(q_1, \ldots, q_s)$ where $1 \le n_0 < m$ and $n_0 = n - tm$, then uses the Algorithm 2 to obtain the final design. Therefore, it is possible to construct the uniform designs even when n is very large. Next, we use some examples to show the efficiency of the combination method.

Example 1. Many uniform design tables in the UD website were constructed by different methods such as number theoretic methods, stochastic optimization and so on. When the number of factors is small, these existing designs can be checked whether their WD-values attain the minimum or not under the U-type constraint. In fact, many existing designs can be constructed by CM. For example, when $q = 3, s = 2$, it can be shown that the frequency vector of different number of runs are as follows

$$\boldsymbol{y}(U_{15}(3^2)) = (2, 2, 1, 2, 1, 2, 1, 2, 2)' = (1, 1, 0, 0, 1, 1, 1, 0, 1)' + \mathbf{1}_m = \boldsymbol{y}_1^1 + \mathbf{1}_m,$$
$$\boldsymbol{y}(U_{24}(3^2)) = (3, 3, 2, 3, 2, 3, 2, 3, 3)' = (1, 1, 0, 1, 0, 1, 0, 1, 1)' + 2\mathbf{1}_m = \boldsymbol{y}_1^2 + 2\mathbf{1}_m,$$
$$\boldsymbol{y}(U_{33}(3^2)) = (3, 4, 4, 4, 4, 3, 4, 3, 4)' = (0, 1, 1, 1, 1, 0, 1, 0, 1)' + 3\mathbf{1}_m = \boldsymbol{y}_1^3 + 3\mathbf{1}_m,$$
$$\boldsymbol{y}(U_{42}(3^2)) = (5, 5, 4, 4, 5, 5, 5, 4, 5)' = (1, 1, 0, 0, 1, 1, 1, 0, 1)' + 4\mathbf{1}_m = \boldsymbol{y}_1^1 + 4\mathbf{1}_m,$$
$$\boldsymbol{y}(U_{51}(3^2)) = (6, 5, 6, 6, 6, 5, 5, 6, 6)' = (1, 0, 1, 1, 1, 0, 0, 1, 1)' + 5\mathbf{1}_m = \boldsymbol{y}_1^4 + 5\mathbf{1}_m.$$

It is easily shown that the designs with the frequency vectors $\boldsymbol{y}_1^1, \boldsymbol{y}_1^2, \boldsymbol{y}_1^3, \boldsymbol{y}_1^4$ have the same WD-value, which means CM can be used to construct these five designs in the same way. Similarly, we can check the existing designs $U_{12}(3^2), U_{21}(3^2), U_{30}(3^2), U_{39}(3^2), U_{48}(3^2)$ also can be constructed by CM, and the same is true for the groups $\{U_8(4^2), U_{24}(4^2), U_{40}(4^2)\}$, $\{U_{12}(4^2), U_{28}(4^2), U_{44}(4^2)\}$, $\{U_{20}(4^2), U_{36}(4^2), U_{52}(4^2)\}$, $\{U_{10}(5^2), U_{35}(5^2)\}$, $\{U_{15}(5^2), U_{40}(5^2)\}$, $\{U_{20}(5^2), U_{45}(5^2)\}$, $\{U_{30}(5^2), U_{55}(5^2)\}$, $\{U_{12}(6^2), U_{48}(6^2)\}$, $\{U_{18}(6^2), U_{54}(6^2)\}$ and $\{U_{24}(6^2), U_{60}(6^2)\}$. For $q = 3, s = 3$, the groups $\{U_9(3^3), U_{36}(3^3)\}$, $\{U_{12}(3^3), U_{39}(3^3)\}$, $\{U_{15}(3^3), U_{42}(3^3)\}$, $\{U_{18}(3^3), U_{45}(3^3)\}$, $\{U_{21}(3^3), U_{48}(3^3)\}$ and $\{U_{24}(3^3), U_{51}(3^3)\}$ also can be constructed by CM.

Example 2. In a similar way to Example 1, we can construct $U_9(3^3), U_{63}(3^3), U_{90}(3^3), U_{117}(3^3)$ by the frequency vectors of $\boldsymbol{y}_1, \boldsymbol{y}_1 + 2\mathbf{1}_m, \boldsymbol{y}_1 + 3\mathbf{1}_m, \boldsymbol{y}_1 + 4\mathbf{1}_m$, respectively, where

$$\boldsymbol{y}_1 = (0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0)'$$

is the frequency vectors of $U_9(3^3)$ and the latter is from the UD website. Therefore, the frequency vector $\boldsymbol{y}_1$ can be used to construct the design $U_n(3^3)$ with any $n = 9 + 27t,\ t \ge 1$. In such a way we can construct many uniform design tables which do not exist in the UD website.

4.2. Empirical study

In this subsection, we compare the uniformity of the design constructed by CM and other methods such as the random permuting method (RPM) and some number theoretic methods including the glp method and the pglp method. Both the symmetrical designs and asymmetrical designs are considered. The procedure of RPM is as follows: to generate a U-type designs $U(n; n^s)$ whose elements of every column are the random permutation of $\{1, \ldots, n\}$, then to transform $U(n; n^s)$ into $U(n; q_1, \ldots, q_s)$

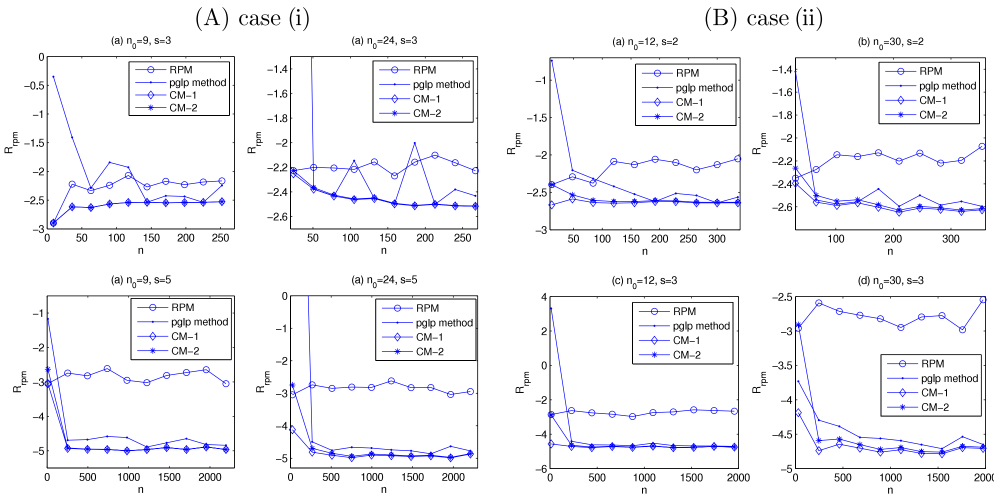

Fig. 2. Comparisons among the RPM, the pglp method, CM-1 and CM-2 for the two cases.

and to calculate the corresponding WD. Repeat this procedure R times and we obtain R WD values, their mean (M_{rpm}), standard deviation (std_{rpm}), and 0.1% percentile of WD values (WD_{rpm}), the latter is denoted as the WD-value of the resulted design from RPM. We take $R = 10^7$ in the simulation.

Let $R_{\text{rpm}} = (\text{WD}_c - M_{\text{rpm}})/\text{Std}_{\text{rpm}}$, where WD_c stands for the WD of any design. Therefore, a design $\boldsymbol{X} \in \mathcal{U}(n; q_1, \ldots, q_s)$ with lower R_{rpm} is more uniform. For studying the robustness of CM, we consider the following two method of CM: CM-1 means the original frequency vector $\boldsymbol{y}_1^*$ is calculated from existing design on UD website or the design constructed by SA-IPM; CM-2 means that the $\boldsymbol{y}_1^*$ is calculated from the design from RPM.

(A) *Symmetrical designs*: The following two cases are considered: (i) the number of levels $q = 3$, the number of factors $s = 3, 5$ and $n_0 = 9, 24$; (ii) $q = 6$, $s = 2, 3$, $n_0 = 12, 30$. For every combination in the two cases, we consider the number of runs $n = n_0 + tq^s$, $t = 0, 1, \ldots, 9$. The comparisons among the RPM, the glp method, CM-1 and CM-2 for the two cases are shown in Fig. 2, where the R_{rpm}-values of different methods are calculated. It is shown that the design constructed by CM-1 has the least WD for every t, and with the increase of t, CM-2 is better than the RPM and the pglp method. Moreover, as t increases, the WD-values of the designs constructed by CM-1 and CM-2 decrease, which means the importance of the uniformity of $U(n_0; q^s)$ decreases when the number of runs increases. Usually, the design constructed by the pglp method has less WD than that by RPM, but from Fig. 2(A) it is seen that this is not true in some cases. Therefore, the CM has some robustness of the uniformity of the design $U_{n_0}(q^s)$, where $0 \le n_0 < m$ and $n_0 = n \pmod m$.

(B) *Asymmetrical designs*: The following four cases are considered: (iii) $s = 3$, $q_1 = q_2 = 3$, $q_3 = 4$; (iv) $s = 3$, $q_1 = q_2 = 3$, $q_3 = 6$; (v) $s = 4$, $q_1 = q_2 = 3$, $q_3 = q_4 = 4$; (vi) $s = 4$, $q_1 = q_2 = 3$, $q_3 = 4$, $q_4 = 6$. In every case, we choose the number of runs $n = 24 + tm$, $t = 0, 1, \ldots, 9$, $m = q_1 \cdots q_s$. Similarly, we compare the property of the RPM, the pglp method, CM-1 and CM-2. Since the designs in these cases do not exist in the UD website, the $\boldsymbol{y}_1^*$ in CM-1 is obtained by SA-IPM. The comparisons among these methods are shown in Fig. 3, from which it is seen that CM-1 is the best one in every case and CM-2 is also better than the RPM and the pglp method when $t \ge 1$. As the number of runs increases, the pglp method is better than RPM. Usually, the property of the pglp method is affected by n, and the R_{rpm} of the pglp method is unstable as n increases, whereas the R_{rpm} of CM-1 and CM-2 are stable, which also means the CM is robust in some sense.

5. Conclusion

In this paper, we reformulate the WD of asymmetrical design as a quadratic form of the frequency vector $\boldsymbol{y}$. The problem of constructing a uniform design under WD then becomes a quadratic

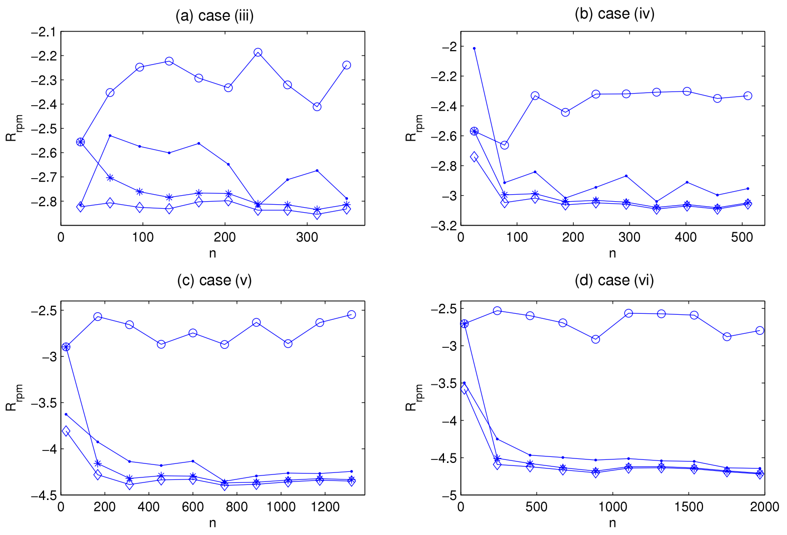

Fig. 3. The comparison of the RPM, the pglp method, CM-1 and CM-2 for asymmetrical designs. Here, "– ○", "– ·", "– ◇", "– ∗" respectively represent the RPM, the pglp method, CM-1 and CM-2.

optimization problem, i.e., the main objective is to find the minimizer of frequency vector. Some theoretic results are obtained. Furthermore, for $n < m$, a heuristic algorithm, SA-IPM, which is based on the simulated annealing, is proposed; for $n > m$, the so-called combination method (CM) is proposed. Our empirical study shows that when $m < 3200$, these algorithms can obtain designs with lower computational complexity and lower WD value compared with many existing construction methods. Therefore, SA-IPM is suitable to construct designs with a large number of runs and the total number of level-combinations to be below several thousands, e.g. $n > 200$ and $m < 3200$. As the value of m increases, the computational complexity of SA-IPM increases exponentially. In the case of the number of runs $n > m$, our empirical study shows that the CM is better than many existing methods.

An interesting result of the algorithm SA-IPM is that the resulted designs are all U-type designs in our empirical tests. Moreover, we also discuss the cases of $\binom{m}{n}$ being not too large, and find that the uniform design among the lattice designs may be a U-type design. This gives additional evidence of using the U-type constraint for construction of UDs. Finally, based on the relationship between constructing uniform designs and the quadratic integer programming, more fast heuristics may be proposed in the future, especially when m is very large.

Acknowledgments

The authors would like to thank the associate editor and the referees for the valuable comments, and Prof. Zhu Yunmin and Dr. Song Enbing for their help on convex optimization and Dr. Zhang Lingyun for his valuable comments. Zhou's work was partially supported by NSFC (11001186). Fang's work was partially supported by the Research Foundation from Academy of Mathematics and System Sciences, Chinese Academy of Sciences. Ning's work was partially supported by NSFC (11101173).

References

[1] M.M. Amini, B. Alidaee, G.A. Kochenberger, A scatter search approach to unconstrained quadratic binary programs, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, 1999, pp. 317–329.
[2] J.E. Beasley, Heuristic algorithms for the unconstrained binary quadratic programming problem, Technical Report, Management School, Imperial College, London, UK, 1998.
[3] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[4] W. Chen, L.S. Zhang, Global optimality conditions for quadratic 0–1 optimization problems, Journal of Global Optimization 46 (2010) 191–206.

[5] K.T. Fang, R.Z. Li, A. Sudjianto, Design and Modeling for Computer Experiments, Chapman and Hall, CRC, New York, 2006.
[6] K.T. Fang, C.X. Ma, Wrap-around L_2-discrepancy of random sampling, Latin hypercube and uniform designs, Journal of Complexity 17 (2001) 408–424.
[7] K.T. Fang, Y. Tang, J.X. Yin, Lower bounds for wrap-around L_2-discrepancy and constructions of symmetrical uniform designs, Journal of Complexity 21 (2005) 757–771.
[8] K.T. Fang, Y. Wang, Number-Theoretic Methods in Statistics, Chapman and Hall, London, 1994.
[9] M. Gilli, P. Winker, Heuristic optimization methods in econometrics, in: D. Belsley, E. Kontoghiorghes (Eds.), Handbook of Computational Econometrics, Wiley, Chichester, 2009, pp. 81–119.
[10] F.J. Hickernell, A generalized discrepancy and quadrature error bound, Mathematics of Computation 67 (1998) 299–322.
[11] F.J. Hickernell, Lattice rules: how well do they measure up? in: P. Hellekalek, G. Larcher (Eds.), Random and Quasi-Random Point Sets, in: Lecture Notes in Statistics, vol. 138, Springer, New York, 1998, pp. 109–166.
[12] R. Horst, P.M. Pardalos, N.V. Thoai, Introduction to Global Optimization, Kluwer, Dordrecht, 1995.
[13] L.D. Iasemidis, P. Pardalos, J.C. Sackellares, D.-S. Shiau, Quadratic binary programming and dynamical system approach to determine the predictability of epileptic seizures, Journal of Combinatorial Optimization 5 (2001) 9–26.
[14] K. Katayama, H. Narihisa, Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem, European Journal of Operational Research 134 (1) (2001) 103–119.
[15] P. Merz, B. Freisleben, Genetic algorithms for binary quadratic programming, in: Proceedings of the 1999 Genetic and Evolutionary Computation Conference, vol. 1, 1999, pp. 417–424.
[16] P. Merz, B. Freisleben, Greedy and local search heuristics for unconstrained binary quadratic programming, Journal of Heuristics 8 (2002) 197–213.
[17] P. Merz, K. Katayama, Memetic algorithms for the unconstrained binary quadratic programming problem, BioSystems 78 (2004) 99–118.
[18] H. Niederreiter, Random number generaion and quasi-Mente Carlo methods, in: SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, 1992.
[19] G. Palubeckis, Multistart tabu search strategies for the unconstrained binary quadratic optimization problem, Annals of Operations Research 131 (2004) 259–282.
[20] P. Winker, K.T. Fang, Application of threshold-accepting to the evaluation of the discrepancy of a set of points, SIAM Journal on Numerical Analysis 34 (1997) 2028–2042.
[21] Y.D. Zhou, J.H. Ning, Lower bounds of the wrap-around L_2-discrepancy and relationships between MLHD and uniform design with a large size, Journal of Statistical Planning and Inference 138 (2008) 2330–2339.