Journal of Statistical Planning and Inference 140 (2010) 169 – 179

Contents lists available at ScienceDirect

Journal of Statistical Planning and Inference

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

Graph based isomorph-free generation of two-level regular fractional factorial designs

Abhishek K. Shrivastava, Yu Ding*

Texas A&M University, Department of Industrial and Systems Engineering, College Station, TX 77843-3131, USA

ARTICLE INFO

Article history:
Received 2 August 2008
Received in revised form
1 July 2009
Accepted 2 July 2009
Available online 9 July 2009

Keywords:
Isomorphism
Graphs
Design catalog
Automorphism
Large designs
Defining contrast subgroup
Treatment combinations subgroup
Canonical labeling

ABSTRACT

We provide a new necessary and sufficient check for testing the isomorphism of two 2-level regular fractional factorial designs. The approach is based on modeling fractional factorial designs as bipartite graphs. We employ an efficient canonical graph labeling approach to compare two designs for isomorphism. We then improve upon the existing non-isomorphic fractional factorial design generation algorithm by reducing the number of candidate designs from which isomorphs need to be removed. Not only does our method generate non-isomorphic designs much faster, it is also able to generate designs with run sizes of 2048 and 4096 runs, which were not generated by the existing methods.

1. Introduction

Fractional factorial designs are a popular choice in designing experiments for studying the effects of multiple factors simultaneously. A good introduction to factorial designs can be found in Wu and Hamada (2000) and Box and Hunter (1961, 2000). Recently, there has been greater interest in designing experiments for studying large-scale systems. The first step in planning such an experiment is the selection of an appropriate fractional factorial design. An appropriate design is one that has the statistical properties of interest of the experimenter and has small number of runs. This requires that a catalog of candidate designs be available (or be possible to generate) for searching for the 'good' design. In the attempt to generate the catalog of candidate designs, the problem of design isomorphism must be addressed.

Two designs with the same number of runs, factors and levels are called equivalent or *isomorphic* to each other if one can be obtained from the other by some relabeling of factor labels, changes in run order or relabeling of level labels in the design matrix. This is also known as *combinatorial isomorphism* in literature (Cheng and Ye, 2004). A catalog of candidate designs should only contain those designs that are distinct or non-isomorphic to each other. This is so because statistical properties of two isomorphic designs are the same. Keeping isomorphic designs wastes the effort of the experimenter when searching for a design. Moreover, the size of the catalog can be greatly reduced by discarding isomorphs. For example, according to Chen et al. (1993), the total number of 2^{15-10} designs is 5,311,735, where the number of unique or non-isomorphic designs of resolution ≥ 3 is only 144.

The objective of this paper is to provide an efficient method for generating all the distinct 2-level regular fractional factorial designs. The typical solution procedure primarily involves discarding isomorphs from the large collection of designs by using some isomorphism check. The problem of checking two designs for isomorphism, however, is combinatorial as the total number

* Corresponding author. Tel.: +1 979 845 5448; fax: +1 979 847 9005.
E-mail addresses: akshriv@tamu.edu (A.K. Shrivastava), yuding@iemail.tamu.edu (Y. Ding).

0378-3758/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.jspi.2009.07.003

Table 1
Isomorphism checks proposed in literature.

	Isomorphism check	Type of check	Type of design	Relevant papers
1	Word length pattern (WLP)	Necessary	Regular, multi-level	Draper and Mitchell (1967), Chen et al. (1993)
2	Letter pattern matrix	Necessary	Regular, multi-level	Draper and Mitchell (1970)
3	Hamming distance based	Necessary and sufficient	General, multi-level	Clark and Dean (2001)
4	Centered L_2 discrepancy	Necessary	General, multi-level	Ma et al. (2001)
5	Extended word length pattern	Necessary	General, 2-level	Sun et al. (2002)
6	Minimal column base	Necessary and sufficient	General, 2-level	Sun et al. (2002)
7	Indicator function representation based	Necessary and sufficient	General, 2-level	Cheng and Ye (2004)
8	Moment projection pattern	Necessary	Regular, multi-level	Xu (2005)
9	Coset pattern matrix	Necessary	Regular, 2-level	Zhu and Zeng (2005)
10	Eigenvalues of word pattern matrices	Necessary (and conjectured sufficient)	Regular, 2-level	Lin and Sitter (2008)

of relabelings of a 2^{n-k} designs is $(n!)(2!)^n(2^{n-k}!)$. Also, in general, the total number of 2^{n-k} designs is $\binom{(2^{n-k}-1)-(n-k)}{k}$. Thus, the problem of generating non-isomorphic designs, given an efficient isomorphism check, involves comparing a combinatorially large number of designs. For these reasons this problem of generating fractional factorial designs is computationally tough. The best results, for generating the collection of non-isomorphic 2-level regular fractional factorial designs, have so far been reported by Lin and Sitter (2008). Their partial check is a fast method for comparing two fractional factorial designs for isomorphism. The partial check is proven necessary but appears to be necessary and sufficient in practice (Lin and Sitter, 2008). The largest catalogs, of non-isomorphic designs, that they could generate were the set of 512-run designs with resolution ≥ 5 and 1024-run even designs with resolution ≥ 6. The generation of 2^{17-8} (512-run) designs with resolution ≥ 5 took them about 12 days (Lin and Sitter, 2008).

We provide a new approach for testing the isomorphism of two 2-level regular fractional factorial designs by modeling them as simple bipartite graphs. The problem is now transformed into a graph isomorphism problem. We use an efficient graph isomorphism check algorithm (McKay, 1981) to provide a necessary and sufficient check for design isomorphism. For generating the complete set of non-isomorphic 2-level fractional factorial designs, we improve Lin and Sitter (2008)'s generation algorithm by using our isomorphism check and reducing the size of the collection of designs from which isomorphs are to be eliminated. We extend some results from graph isomorphism literature to develop this reduction procedure.

The contributions of our paper are twofold: (i) we provide a new necessary and sufficient check for design isomorphism; (ii) we provide a design generation algorithm that can generate non-isomorphic designs much faster than any of the previous methods. As a result of this we are able to generate designs with run sizes of 2048 and 4096 runs, which were not generated by any existing methods.

The rest of this paper is organized as follows. Section 2 surveys the literature on fractional factorial design isomorphism. In Section 3, we give details on the proposed isomorphism check. Next, we give details on the isomorph-free design generation algorithm in Section 4. Section 5 presents the computational results using the proposed design generation algorithm and compares them with other methods. We conclude in Section 6, giving insights into the improvements and possible extensions of the current work.

2. Literature review

There are two main components in the procedure for generating fractional factorial designs—the isomorphism check and the design generation algorithm. The isomorphism check gives a condition that can be used to test if two designs are isomorphic or not. The design generation algorithm provides a procedure to generate the entire non-isomorphic set without considering all possible comparisons among all the designs.

2.1. Isomorphism checks

Isomorphism checks can be categorized by their classification capability—whether they are necessary or, both, necessary and sufficient[1] conditions. Necessary checks are usually faster than necessary and sufficient checks but are not always able to differentiate between two isomorphic designs. Table 1 summarizes the major isomorphism checks in literature. We do not claim that we have listed all the approaches proposed in literature but we do list the most promising approaches proposed for regular fractional factorial designs. For a comprehensive review, see Katsaounis and Dean (2008).

Most of the criteria summarized in Table 1 provide necessary conditions only, except Clark and Dean (2001), Sun et al. (2002) and Cheng and Ye (2004), who give a necessary and sufficient check, and possibly Lin and Sitter (2008), who conjecture that their proposed check may be sufficient also. Ma et al. (2001) extended Clark and Dean (2001)'s work and proposed a necessary check, which they conjectured as sufficient, by using the Hamming distance matrix to define the centered L_2-discrepancy, a uniformity measure, between two designs. We found that Ma et al. (2001)'s sufficiency conjecture does not hold true for 2-level

[1] We do not have a sufficient (and not necessary) category since we did not find any isomorphism checks that would fall in this category.

regular fractional factorial designs (see Appendix B for counterexample). It may be noted that, among the checks listed in Table 1, Clark and Dean (2001) and Ma et al. (2001) are applicable to general (regular and non-regular) multi-level fractional factorial designs. Sun et al. (2002) and Cheng and Ye (2004) are also applicable to non-regular designs but are limited to 2-level designs for checking for (combinatorial) isomorphism. Xu (2005)'s check is applicable to multi-level designs but is limited to regular fractional factorial designs. Zhu and Zeng (2005) and Lin and Sitter (2008)'s checks are restricted to the class of 2-level regular fractional factorial designs.

It is also essential to note that there is a fundamental difference between the working of the eigenvalue check (Lin and Sitter, 2008) and the other necessary and sufficient checks. For comparing a collection of m designs, the eigenvalue check involves running m expensive computations, one for each design and then comparing pairs of eigenvalues (which is computationally cheap). The other methods, on the other hand, run the expensive computations on each pair of designs to determine if they are isomorphic or not, which is $m(m-1)/2$ runs, in the worst case. This fundamental difference makes the eigenvalue check more attractive for use in a design generation algorithm.

2.2. Design generation algorithm

The trivial way of finding all the non-isomorphic designs for a given number of factors, n, and fraction, k (or equivalently, the number of runs, 2^{n-k}) is to generate all the 2^{n-k} designs and compare them for isomorphism. But this approach is impractical due to the excessively large number of designs (even for small n and k) and the costly isomorphism check. The approaches in Chen et al. (1993), Xu (2005), and Lin and Sitter (2008) reduce the computational burden by generating a subset of the entire set of 2^{n-k} designs. Chen et al. (1993) generated this subset by constructing 2^{n-k} designs by adding defining words only to non-isomorphic $2^{(n-1)-(k-1)}$ designs. Bingham and Sitter (1999) further reduced this set by allowing only those defining words to be used that follow certain ordering. Chen et al. (1993), Xu (2005), and Lin and Sitter (2008) then decompose this set of designs into smaller subsets, using a necessary condition (e.g., word length pattern in Chen et al., 1993, or eigenvalue criterion in Lin and Sitter, 2008, among others), such that two designs belonging to different subsets are non-isomorphic to each other. Isomorphs within each subset are then discarded using some necessary and sufficient check.

3. The isomorphism check

A 2-level regular fractional factorial design, 2^{n-k}, has n factors, each with two levels, and consists of 2^a ($a = n-k$) runs. Thus, it is the $1/2^k$th fraction of a 2^n full factorial design, where the fraction is determined by k *defining words*. Each *word* consists of letters (e.g., A, B, C, D) denoting factors. Each letter, denoting a factor, is an element of Galois field $GF(2)$. The defining words form an abelian group called the *defining contrast subgroup*. The defining words are a set of generators for this group. The group, therefore, consists of 2^k words, including the identity element which is usually denoted by I.

A fractional factorial design is usually presented in a table, the experimental plan, listing the (ordered) runs and the factor levels at each run. A design table (or design matrix) is similar to the experiment table but without any specific ordering of the runs. A 2-level regular fractional factorial design table is uniquely defined by the number of factors, n, and its defining contrast subgroup, S (or equivalently, a set of defining words). We denote this representation by the tuple $\{n, S\}$. It must be noted that the defining words are not unique for a given design, i.e., two distinct sets of defining words may exist that generate the same defining contrast subgroup.

3.1. 2-Level regular fractional factorial design isomorphism problem

Since a design matrix is uniquely defined by its defining contrast subgroup, we have Lemma 3.1, which essentially paraphrases Theorem 5 of Chen (1992). Hence, we omit its proof.

Lemma 3.1. *Two 2-level regular fractional designs, $d_1 \equiv \{n, S_1\}$ and $d_2 \equiv \{n, S_2\}$, are isomorphic to each other if and only if one of S_1 or S_2 can be obtained from the other by some permutation of factor labels and reordering of words.*

Lemma 3.1 provides a way of testing if two designs are isomorphic by comparing their defining contrast subgroups for isomorphism. To solve this, we introduce a graph representation for the defining contrast subgroup.

3.2. 2-Level regular fractional factorial designs as graphs

A simple undirected *graph* $G(V, E)$ consists of disjoint finite sets V of *vertices*, and E of *edges*. Each edge is a pair of distinct vertices, and no two edges repeat in the edge set E. We provide a new bipartite graph representation of a 2-level regular fractional factorial design. A bipartite graph $G(V_a, V_b, E)$ is a graph in which the vertex set V can be partitioned into disjoint subsets V_a and V_b such that each edge has one vertex in V_a and one in V_b.

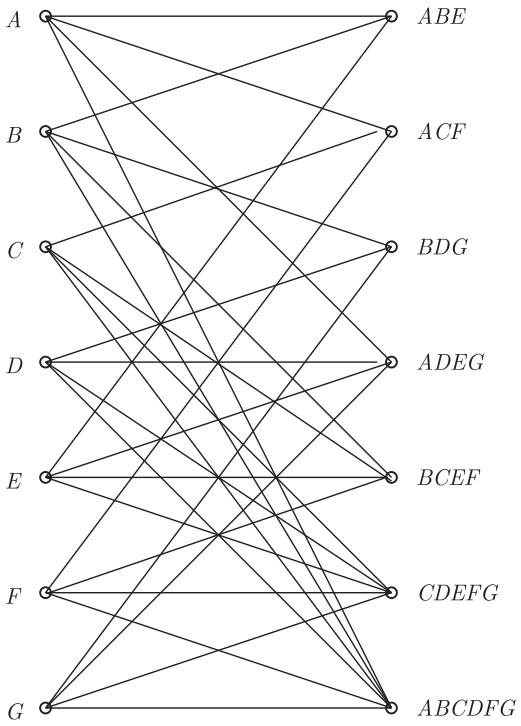

Fig. 1. Bipartite graph for the 2^{7-3} design with defining contrast subgroup $\{ABE, ACF, BDG, ADEG, BCEF, CDEFG, ABCDFG\}$. Vertices on the left, set V_a, correspond to factors, and vertices on the right, set V_b, correspond to words in the contrast subgroup.

Algorithm 1. *Construction of bipartite graph $G(V_a, V_b, E)$ for design $d \equiv \{n, S\}$*
Input: *design $d \equiv \{n, S\}$*

(1) *Start with an empty graph with no vertices, i.e., $V_a = \phi$ and $V_b = \phi$ (and hence, no edges, i.e., $E = \phi$).*
(2) *For each factor in the design d, add a vertex in V_a, i.e., add vertices $v_{a1}, \ldots, v_{an}$ in V_a.*
(3) *For each word in the defining contrast subgroup S, except I, add a vertex in V_b, i.e., add vertices $v_{b1}, \ldots, v_{b(|S|-1)}$ in V_b, where $|S|$ denotes the cardinality of set S.*
(4) *For each word in S, except I, add edges between the vertex (in V_b), corresponding to the word, and the vertices (in V_a), corresponding to the factors in the word.*

Fig. 1shows the graph representation of the 2^{7-3} design with contrast subgroup $\{I, ABE, ACF, BDG, ADEG, BCEF, CDEFG, ABCDFG\}$. The vertex *ABE* in V_b, for example, is connected by edges to vertices *A*, *B*, and *E* in V_a.

An alternate graph representation of the design can be obtained by using the treatment combination subgroup, *T*, of the design instead of the defining contrast subgroup, *S*. The treatment combination subgroup of a design is the group formed by the runs in the design matrix. For example, for the 2^{7-3} design in Fig. 1, the treatment combination group consists of 16 elements $\{I, DG, CF, CDFG, BEG, BDE, BCEFG, BCDEF, AEF, ADEFG, ACE, ACDEG, ABFG, ABDF, ABCG, ABCD\}$, each corresponding to the runs in the principal fraction. Since the relationship between the defining contrast subgroup and the treatment combination subgroup is well known to be one-to-one (see for instance Bailey, 1977), it is clear that Lemma 3.1 holds when the defining contrast subgroup is replaced with the treatment combination subgroup in the lemma. The alternate graph representation for a design $\{n, T\}$ is also obtained by following Algorithm 1, but with *S* replaced with *T*. For fixed *n*, since the size of the constructed graph depends on the size of *S* or *T*, the alternate representation gives a smaller graph whenever $|T| = 2^{n-k} < 2^k = |S|$ or $n < 2k$. Thus, when converting a design to a graph, we may choose one of the two representations depending on whether $n < 2k$ or not.

3.3. Graph isomorphism and fractional factorial design isomorphism

The graph isomorphism problem is to check, given two graphs, if there exists a relabeling of the vertices of one graph that would make it identical to the other. The relabeling should preserve the vertex adjacency of the vertices, i.e., if vertices v_1 and v_2 have an edge between them then the relabeled vertices v_1' and v_2', respectively, should also have an edge between them. Fig. 2 shows an example of isomorphic graphs. The graphs in Fig. 2 are essentially mirror images of each other, and one can be obtained from the other by exchanging vertex labels *A* with *B* and *C* with *D*.

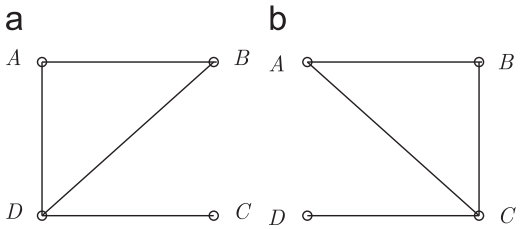

Fig. 2. (a) and (b) show two simple graphs isomorphic to each other. Exchanging vertex labels A with B and C with D gives the other graph.

The fractional factorial design isomorphism problem can be translated to the problem of checking isomorphism between the corresponding graph representations of the two designs. The relabelings of factors of a design then correspond to the permutations of vertex labels that preserve the partitions and vertex adjacencies in the graph. This is the set of all vertex adjacency preserving permutations that allow permutations only within each of the partitions.

Theorem 3.2. *Two 2-level regular fractional factorial designs, $d_1 \equiv \{n, S_1\}$ and $d_2 \equiv \{n, S_2\}$, with graph representations $G_1(V_{a1}, V_{b1}, E_1)$ and $G_2(V_{a2}, V_{b2}, E_2)$, respectively, are isomorphic to each other if and only if G_1 and G_2 are isomorphic to each other.*

Proof. See Appendix A.

Theorem 3.2 gives a *necessary and sufficient condition* for checking the isomorphism between two fractional factorial designs by solving the graph isomorphism problem. It is clear that a version of Theorem 3.2 can also be written for the case when the design is represented by its treatment combination subgroup. Since the extension is straightforward we skip the details here. Our approach to solving the graph isomorphism problem is discussed in Section 3.4.

3.4. Solving the graph isomorphism problem

The graph isomorphism problem has been extensively studied in mathematics and computer science. Much effort has been put in developing efficient algorithms for this problem. For a review on the history of the problem and algorithmic developments towards solving this problem, see Read and Corneil (1977) and Fortin (1996). The graph isomorphism problem has its special place in complexity theory. It is known to be in NP, but it is not known whether it is in P or NP-complete. The graph construction for a design, as given in Algorithm 1, has time complexity $O(n \cdot |S|)$ (all steps other than step (4) of the construction take linear time in n or $|S|$, the number of words in S), or $O(n \cdot |T|)$, if the treatment combination subgroup is used instead. Thus, a design can be transformed into a graph in polynomial time (polynomial in the number of factors and the number of words in S or T). Therefore, the problem of determining whether two designs are isomorphic or not is no tougher than the graph isomorphism problem.

There are two primary approaches to solving the graph isomorphism problem. One is to test the isomorphism between two graphs by directly attempting to find a relabeling map that makes one graph identical to the other. The other approach, which we use, is the so-called *canonical labeling* approach. In this approach, a canonical graph is obtained for each graph. The canonical graph is computed in such a way that if two graphs are isomorphic to each other, then the canonical graphs computed for the two graphs are identical.

The most efficient canonical labeling algorithm is implemented in a C package *nauty* based on McKay (1981). This package is available freely for research purposes from the developer's website (McKay, 2004). The algorithm is known to take exponential running time, in the number of vertices, in the worst case (Kocay, 1996). This suggests that, in the worst case, the design isomorphism problem can be solved in exponential time in the number of words in the defining contrast subgroup or treatment combination subgroup (since Algorithm 1 requires $O(n \cdot |S|)$ or $O(n \cdot |T|)$ running time to transform a design to a graph). But in practice *nauty* has been found to be extremely efficient for most graphs and outperforms all other graph isomorphism algorithms (Kocay, 1996). Therefore, we expect our isomorphism check to also be very efficient for most 2-level regular fractional factorial designs.

4. Generating isomorph-free collections of 2-level fractional factorial designs

We generate the set of non-isomorphic 2^{n-k} designs in a recursive manner, as in Chen et al. (1993) and Lin and Sitter (2008). We start with the only 2^a ($a = n - k$) full factorial design, generate all non-isomorphic $2^{(a+1)-1}$ designs, then all non-isomorphic $2^{(a+2)-2}$ designs, …, and finally all non-isomorphic $2^{(a+k)-k}$ (i.e., 2^{n-k}) designs. Each intermediate step involves constructing larger (children) designs by adding a new factor and defining word to smaller (parent) designs.

For generating the set of non-isomorphic $2^{(n+1)-(k+1)}$ designs from the set of 2^{n-k} designs, we propose an algorithm similar to Lin and Sitter (2008). This algorithm differs from Lin and Sitter (2008) in two ways. The first difference is the choice of the isomorphism check used. We replace their eigenvalue check with our graph-based check. The second difference is the added

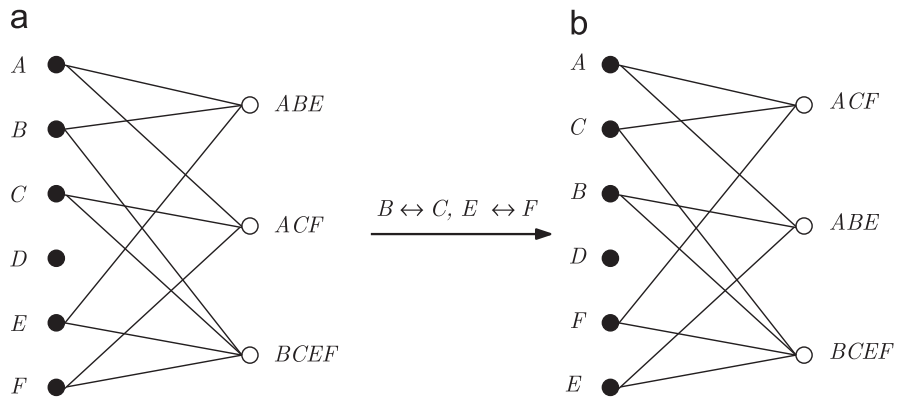

Fig. 3. (a) is the graph representation of the 2^{6-2} design with defining contrast subgroup $\{ABE, ACF, BCEF\}$. The relabeling $B \leftrightarrow C$ and $E \leftrightarrow F$ is an automorphism of this design as the graph in (b) is identical to that in (a).

candidate word reduction method, which reduces the number of candidate designs generated from which isomorphs need to be removed. We first describe the generation algorithm proposed by Lin and Sitter (2008) in Section 4.1, followed by the candidate word reduction method in Section 4.2 and then finally give our proposed algorithm in Section 4.3.

4.1. The basic algorithm

The basic algorithm proceeds by first constructing the set of all candidate words C. Candidate words are defining words constructed from the first a $(=n-k)$ factors. The set C is an ordered set, ordered first by word lengths and then by lexicographic ordering to break the ties. Then, candidate $2^{(n+1)-(k+1)}$ designs are constructed by adding a candidate defining word to each design d in $D_{n,k}$, the set of all non-isomorphic 2^{n-k} designs. Those candidate designs for which the last added defining word (when constructing the 2^{n-k} design from a $2^{(n-1)-(k-1)}$ design) lies before the newly added defining word are not allowed (Lin and Sitter, 2008; Bingham and Sitter, 1999). These candidate designs make up the set $D^{+}_{n+1,k+1}$. This set is then partitioned into subsets $G_1, \ldots, G_m$ using a necessary isomorphism check. We use the word length pattern check, also used in Lin and Sitter (2008), in our implementation, as it is computationally inexpensive. We then use our graph based isomorphism check to remove the isomorphs from each subset. The subsets, together, now form the set $D_{n+1,k+1}$, the set of non-isomorphic $2^{(n+1)-(k+1)}$ designs.

Since the basic algorithm described above is similar to that in Lin and Sitter (2008) and Bingham and Sitter (1999), except for the graph based isomorphism check, we skip the proof that the algorithm actually finds all the non-isomorphic designs. The algorithm reduces the number of 2^{n-k} designs considered for finding the non-isomorphic designs, i.e., the set $D^{+}_{n+1,k+1}$ is smaller than the set of all possible $2^{(n+1)-(k+1)}$ designs. It seems obvious that the smaller the set $D^{+}_{n+1,k+1}$ is the faster the algorithm is going to work. A method for further reducing $D^{+}_{n+1,k+1}$ is described in the next subsection.

4.2. Reducing candidate designs

In this section, we extend an idea suggested by McKay (1998), which proposes an algorithm for generating non-isomorphic graphs, to reduce the candidate words in C. Before we present the main result, we first extend the concept of automorphisms of a graph (Cameron and Mary, 2004) to automorphisms of fractional factorial designs.

Definition 4.1. An automorphism of a design $d \equiv \{n, S\}$ is a relabeling of factor labels of d, such that the design obtained after relabeling is identically d.

Fig. 3(a) shows the graph representation of a 2^{6-2} design with defining contrast subgroup $\{ABE, ACF, BCEF\}$. Fig. 3(b) shows the graph obtained by relabeling factors B with C and E with F. Clearly, the two graphs, and hence the designs, are identical. Therefore, the relabeling—B with C and E with F—is an automorphism of the 2^{6-2} design in Fig. 3.

Theorem 4.2. *Suppose $d \equiv \{n, S\}$ is a parent design, and c_1 and c_2 (not identically equal to c_1) are two candidate defining words. Further suppose that there exists an automorphism α of d, such that c_1 is isomorphic to c_2 under this factor relabeling α. Then, the child designs $d \cup c_1$ and $d \cup c_2$, obtained by adding the defining words c_1 and c_2 to d, respectively, are isomorphic to each other.*

Proof. See Appendix A.

As an example, consider the 2^{6-2} design in Fig. 3. Among the many different possible defining words, consider the two candidate defining words BDG and CDG to be added to the 2^{6-2} design. Under the relabeling $B \leftrightarrow C$ and $E \leftrightarrow F$, which is an

automorphism of the 2^{6-2} design in Fig. 3, *BDG* and *CDG* are clearly isomorphic to each other. The designs obtained from adding *BDG* (Fig. 1) and *CDG* are isomorphic to each other.

The result in Theorem 4.2 allows us to reduce the candidate defining words in C to C' by keeping only the defining words that are non-isomorphic under all the automorphisms of the parent design. That is, for each design we compute all the automorphisms and then reduce the set C to C' by keeping only non-isomorphic defining words. For obtaining the automorphisms of a design, we compute the automorphisms of the corresponding graph representation, which we obtain using *nauty*.

4.3. Proposed enumeration algorithm

Algorithm 2 summarizes our enumeration algorithm by combining the graph based candidate word reduction method of Section 4.2 with the basic algorithm described in Section 4.1.

Algorithm 2. *Generating non-isomorphic* $2^{(n+1)-(k+1)}$ *designs from non-isomorphic* 2^{n-k} *designs*
Input: $D_{n,k}$, *set of all non-isomorphic* 2^{n-k} *designs*

(1) *Construct all possible* $2^a - 1$ *words, except I, from the first* $a = n - k$ *factors, and order them first by their word lengths and then break ties by lexicographic ordering. Call this ordered set C.*
(2) *For each design* $d \in D_{n,k}$
  (a) *Find the set C′, of unique defining words, under the action of the automorphisms of d on C.*
  (b) *Construct a set of* $2^{(n+1)-(k+1)}$ *designs by adding to d a defining word* $c \in C'$, *where c lies below the last added word in the ordered set C.*
(3) *Combining all the designs constructed for each d, form the set* $D^+_{n+1,k+1}$, *the set of candidate designs.*
(4) *Partition the set* $D^+_{n+1,k+1}$ *into subsets* $G_1, \ldots, G_m$, *such that designs in each subset have the same word length pattern but designs in different subsets have distinct word length pattern.*
(5) *Use the graph based isomorphism check of* Section 3 *to compare designs within each subset* G_i, $i = 1, \ldots, m$, *to remove isomorphs from each subset.*
(6) *Collect all the remaining designs* (*in these subsets*) *in* $D_{n+1,k+1}$, *the set of non-isomorphic* $2^{(n+1)-(k+1)}$ *designs.*

In step (5) of Algorithm 2, we construct the graphs for the designs either from their defining contrast subgroups or the treatment combination subgroups depending on whether $n \geq 2k$ or not. Since the designs are generated recursively, starting from the full factorial design, the first few iterations (while $n \geq 2k$) use defining contrast subgroup to construct the graph. Once $n < 2k$ (so that $|S| > |T|$), the treatment combination subgroup is used to construct the graph. Thus, the size of the graph does not increase exponentially (in multiples of 2) forever with each iteration (as n and k increase) but only linearly (in n).

Theorem 4.3. Algorithm 2 *generates the complete set of non-isomorphic* $2^{(n+1)-(k+1)}$ *designs.*

Proof. The proof follows from the basic algorithm and Theorem 4.2. We skip the details here. □

5. Results

Using Algorithm 2 we were able to generate all the designs generated by Chen et al. (1993) and Lin and Sitter (2008). Additionally, we could generate all of 1024-run (resolution ≥ 6), 2048-run (resolution ≥ 7) and 4096-run (resolution ≥ 8) designs. The designs are available for download from the corresponding author's website (http://ise.tamu.edu/metrology/, click on `Publications`). Table 2 shows the number of non-isomorphic designs generated by our algorithm. The numbers in the table match with those in Chen et al. (1993) and Lin and Sitter (2008). We do not present here the catalog of designs. But these can be obtained from the authors.

We compare the computational efficiency of our method with three methods—Clark and Dean (2001)'s algorithm for defining contrast subgroup (DeseqCS), Ma et al. (2001)'s algorithm (MaCD2) and Lin and Sitter (2008)'s eigenvalue check (EigVal). All the computations were done on a Windows Server 2003 R2 Standard x64 edition with an Intel Xeon 3 GHz processor and 16 GB RAM. The programs were written in C++ and built as 32-bit applications with the Microsoft Visual C++ 8.0 compiler. Our implementation of Lin and Sitter (2008)'s eigenvalue check uses LAPACK++ (Stimming, 2007), a C++ library for high performance linear algebra computations, that uses LAPACK (Anderson et al., 1999) and BLAS (Lawson et al., 1979) libraries. Since our implementations may not be most efficient, we will not be interested in small differences in performance.

Tables 3–5 show the cumulative run times for DeseqCS, MaCD2, EigVal, our graph-based algorithm without the candidate reduction step (GBAnoR), and our graph-based algorithm with the candidate reduction step (GBA) for generating 128-, 256- and 512-run designs, respectively. These run times include the time needed to generate a design through the recursive procedure starting from the full factorial design. Between DeseqCS, MaCD2 and EigVal, the run times for EigVal are the best in all cases. Compared to EigVal, for $k \geq 3$, the run times for GBAnoR are reduced by over 95% for 128-, 256-, and 512-run designs. Since the only difference between EigVal and GBAnoR is the isomorphism check used, these large differences indicate that our isomorphism check is significantly faster than the eigenvalue check in Lin and Sitter (2008). Better yet, our check is proven to be necessary and sufficient whereas theirs is only proven necessary. The improvement in run times by including the candidate reduction method

Table 2
Number of non-isomorphic designs by run size.

n	Run size (resolution $\geq r$)								
	16(3)	32(3)	64(3)	128(4)	256(5)	512(5)	1024(6)	2048(7)	4096(8)
5	3	1	*	*	*	*	*	*	*
6	4	4	1	*	*	*	*	*	*
7	5	8	5	1	*	*	*	*	*
8	6	15	14	5	1	*	*	*	*
9	5	29	38	13	5	1	*	*	*
10	4	46	105	33	9	6	1	*	*
11	3	64	273	92	11	16	6	1	*
12	2	89	700	249	14	36	14	6	1
13	1	112	1794	623	15	92	24	9	6
14	1	128	4579	1535	11	282	47	7	7
15	1	144	11,635	3522	6	1011	98	7	4
16	*	145	29,091	7500	1	4019	185	7	5
17	*	129	**	14,438	1	13,759	380	3	5
18	*	113	**	25,064	*	**	919	2	2
19	*	91	**	**	*	**	1701	1	1
20	*	67	**	**	*	**	1682	1	1

A '*' means that no such designs exist. A '**' means that no existing algorithms, including ours, can return a valid result for such a problem size.

Table 3
Comparison of cumulative CPU time (in s) for generating 128-run ($R \geq 4$) designs.

$n-k$	DeseqCS	MaCD2	EigVal	GBAnoR	GBA
8 – 1	0.078	0.297	0.062	0	0
9 – 2	0.484	1.609	0.249	0.015	0
10 – 3	5.484	9.437	1.843	0.046	0.015
11 – 4	55.109	(54.014)	12.484	0.218	0.125
12 – 5	911.421	(307.742)	84.029	1.109	0.671
13 – 6	14,322.5	(1539.8)	523.646	5.39	3.531
14 – 7	**	(6808.46)	3290.970	25.765	18.484
15 – 8	**	(27,747.5)	21,401.300	73.719	57.219
16 – 9	**	**	(9.3 days)	211.362	175.752

The values in parenthesis for MaCD2 are the cases in which MaCD2 did not detect all non-isomorphic designs. The values in parenthesis for EigVal are the ones reported in Lin and Sitter (2008); for the corresponding case our implementation of their algorithm could not handle the problem size. A '**' means that the problem size is too large for our implementation of the corresponding algorithm to give valid results.

Table 4
Comparison of cumulative CPU time (in s) for generating 256-run ($R \geq 5$) designs.

$n-k$	DeseqCS	MaCD2	EigVal	GBAnoR	GBA
9 – 1	0.156	2.25	0.046	0.015	0
10 – 2	1.171	9.296	0.312	0.031	0.015
11 – 3	7.046	37.453	1.906	0.078	0.031
12 – 4	44.25	110.11	6.609	0.203	0.093
13 – 5	179.75	260.05	17.671	0.484	0.265
14 – 6	486.593	407.052	31.530	0.921	0.546
15 – 7	941.046	486.084	41.467	1.359	0.921
16 – 8	1025.16	492.99	42.858	1.656	1.203
17 – 9	1025.34	493.178	43.061	1.843	1.296

is much less but is still impressive. For $k \geq 3$, the run times for GBA are between 30% and 80% of the run times for GBAnoR. Compared to EigVal, the total reduction in run times is over 98% in most cases.

The improvement due to candidate reduction is better reflected in Table 6. The number of designs left in $D^+_{n,k}$, from which isomorphs need to deleted, is about 4000–8000 fewer for 128-run designs with $k \geq 7$, when candidate reduction is used. This is a 16–28% reduction in the number of designs in $D^+_{n,k}$. For larger designs, for which the calls to *nauty* could be more expensive, such reductions may lead to considerable reduction in computation times.

6. Conclusion and discussion

We develop a new necessary and sufficient check for testing the isomorphism of two 2-level fractional factorial designs based on a graph representation of the design. This isomorphism check differs from other necessary and sufficient checks (Chen et al., 1993; Clark and Dean, 2001) in that it does not directly compare two designs. Instead, the method generates a canonical representation of a design such that two isomorphic designs always have the same canonical representation. Our comparisons

Table 5
Comparison of cumulative CPU time (in s) for generating 512-run ($R \geq 5$) designs.

$n-k$	DeseqCS	MaCD2	EigVal	GBAnoR	GBA
10 – 1	0.562	23.469	0.203	0.015	0.015
11 – 2	6.046	119.829	2.109	0.093	0.031
12 – 3	85.265	741.353	20.155	0.484	0.14
13 – 4	912.046	(3863.61)	126.341	2.265	0.75
14 – 5	13,683.7	(19,168.6)	750.344	11.047	5.453
15 – 6	**	(89,653.8)	5119.450	57.219	38.641
16 – 7	**	(100 h)	(30 h)	320.91	271.534
17 – 8	**	**	(12 days)	1877.18	1796.54

The values in parenthesis for MaCD2 are the cases in which MaCD2 did not detect all non-isomorphic designs. The values in parenthesis for EigVal are the ones reported in Lin and Sitter (2008); for the corresponding case our implementation of their algorithm could not handle the problem size. A '**' means that the problem size is too large for our implementation of the corresponding algorithm to give valid results.

Table 6
Number of designs in $D^{+}_{n,k}$ before discarding isomorphs: comparison between our graph-based algorithm with no candidate reduction (GBAnoR) and our graph-based algorithm with candidate reduction (GBA).

k	128 run ($R \geq 4$)		256 run ($R \geq 5$)		512 run ($R \geq 5$)	
	GBAnoR	GBA	GBAnoR	GBA	GBAnoR	GBA
1	98	98	162	162	381	381
2	185	62	227	68	703	166
3	495	177	409	146	2063	496
4	1273	703	480	206	4739	1497
5	3346	2026	453	267	11,077	5731
6	7560	4952	205	137	25,913	18,444
7	15,336	11,110	51	42	60,545	52,917
8	28,766	22,572	2	2	132,909	128,292
9	49,708	41,421	0	0	*	*

A '*' means that the problem size is too large and it prevents our algorithm to return a valid result.

indicate that our proposed isomorphism check runs significantly faster than Lin and Sitter (2008)'s (necessary and possibly sufficient) eigenvalue check, Clark and Dean (2001)'s (necessary and sufficient) method for defining contrast subgroup and Ma et al. (2001)'s (necessary) isomorphism check.

The other necessary and sufficient checks, proposed in Sun et al. (2002) and Cheng and Ye (2004), compare each pair of designs, similar to Clark and Dean (2001)'s method, to determine if they are isomorphic or not. Even if these methods are faster, we do not expect them to have better performance than our isomorphism check for large n. This is because the collection of designs from which isomorphs are to be removed rapidly increases with n. If m is the size of one such collection of designs, then these methods would require $m(m-1)/2$ expensive computations compared to m for our isomorphism check. Moreover, Katsaounis and Dean (2008) compared Cheng and Ye (2004)'s method and Clark and Dean (2001)'s method (which seems to be much slower than our method), among other methods, but did not find enough evidence to conclude that Cheng and Ye (2004)'s method should be faster than Clark and Dean (2001)'s method.

Further, our graph representation allows us to extend results in non-isomorphic graph generation literature to the non-isomorphic design generation problem. Using results from the graph isomorphism problem we improve the existing design generation algorithm of Lin and Sitter (2008) by reducing the number of designs to be tested for isomorphism. We use this algorithm to generate 2-level designs for run sizes up to 4096 and give comparisons of the computational effort. The computational results indicate remarkable improvement in run times and ability to handle large designs, compared to Lin and Sitter (2008).

For Lin and Sitter (2008), it should be noted that even if the eigenvalue check is also sufficient to check the isomorphism of two fractional factorial designs, it may run into a potential problem due to the unavoidable round off errors in floating point computation. The eigenvalues of the matrices constructed in Lin and Sitter (2008)'s method may not all be integers so computing eigenvalues must involve floating point computations, and the eigenvalues computed need to be rounded off. Rounding off eigenvalues may lead to declaring isomorphic designs as non-isomorphic. It is not clear as to what round off level should be chosen and it is not clear, either, how serious this problem could be. Our experience in using the eigenvalue check has turned out positive. We rounded off the eigenvalues to the nearest integer and this worked flawlessly in our implementation. It is also interesting to note that we did not find a single case where the eigenvalue check failed to distinguish between two isomorphic designs.

Further extensions of our isomorphism check and the design generation algorithm to 2-level fractional factorial split plot designs are trivial consequence. The graph constructed in this case is a colored bipartite graph. The vertices corresponding to the factors will need to be partitioned into two sets—one corresponding to the whole-plot factors and the other corresponding to the sub-plot factors. Each set of vertices has a distinct color. The graph isomorphism, in this case, also needs to preserve the colors of the vertices. The extensions to regular multi-level and non-regular designs are non-trivial. They are part of our future work.

Table 7
Distribution of CD_2^2 values.

F_j	CD_2^2	Freq.
F_1	−0.984375	10
F_2	−0.992188	45
F_3	−0.996094	116
	−0.992188	4
F_4	−0.998047	174
	−0.996094	36
F_5	−0.999023	120
	−0.998047	128
	−0.996094	4
F_6	−0.999023	163
	−0.998047	47
F_7	−0.999023	110
	−0.998047	10
F_8	−0.999023	44
F_9	−0.999023	10
F_{10}	−0.999023	1

Appendix A. Proofs of theorems

Proof of Theorem 3.2. First, assume that d_1 and d_2 are isomorphic. Then $\exists$ a permutation α of factor labels such that $S_1^\alpha = S_2$, i.e. S_1 is isomorphic to S_2 (subscript) under the action of α. Consider some word $w_1 = f_{1,1} \ldots f_{m,1} \in S_1$, where $f_{1,1}, \ldots, f_{m,1}$ denote factor labels in d_1. Then $\exists$ a word $w_2 = f_{1,2} \ldots f_{m,2} \in S_2$, where $f_{1,2}, \ldots, f_{m,2}$ denote factor labels in d_2, such that $w_1^\alpha = w_2$ ($\because S_1^\alpha = S_2$). Therefore, the edges in G_1^α are of the form $\{f_{1,1}^\alpha, w_1^\alpha\}$, where $f_{1,1} \in V_{a1}^\alpha$ and $w_1 \in V_{b1}^\alpha$, corresponding to the edges $\{f_{1,2}, w_2\}$, where $f_{1,2} \in V_{a2}$ and $w_2 \in V_{b2}$ in G_2. Since $V_{a1}^\alpha = V_{a2}$ and $V_{b1}^\alpha = V_{b2}$, G_1 is isomorphic to G_2.

Now, assume that G_1 and G_2 are isomorphic, with $V_{a1}^\alpha = V_{a2}$ and $V_{b1}^\beta = V_{b2}$. Let $w_1 = f_{1,1} \ldots f_{m,1} \in S_1$. Let $v_{b1} \in V_{b1}$ correspond to w_1, and $v_{1,a1}, \ldots, v_{m,a1}$ correspond to $f_{1,1}, \ldots, f_{m,1}$, respectively. Similarly, let $w_2 = f_{1,2} \ldots f_{m,2} \in S_2$, $v_{b2} \in V_{b2}$ correspond to w_2, and $v_{1,a2}, \ldots, v_{m,a2}$ correspond to $f_{1,2}, \ldots, f_{m,2}$, respectively. Since, $G_1^{\alpha\beta} = G_2$, we have for edges $\{v_{1,a1}^\alpha, v_{b1}^\beta\} = \{v_{1,a2}, v_{b2}\}$. Since the choice of $v_{1,a1}$ was arbitrary in the last statement, we have, for the corresponding words, $w_1^{\alpha\beta} = w_2$. Again, since the choice of $w_1 \in S_1$ was arbitrary, we have S_1 isomorphic to S_2. Therefore, d_1 is isomorphic to d_2. □

Proof of Theorem 4.2. Since $d \equiv \{n, S\}$, we have $d \cup c_1 = \{n+1, \{S, c_1 S\}\}$ and $d \cup c_2 = \{n+1, \{S, c_2 S\}\}$, where $\{S, c_i S\}$ is the defining contrast subgroup of $d \cup c_i$, $i = 1, 2$. Since α is an automorphism of d, we only need to show that $(c_1 S)^\alpha = c_2 S$ to prove that $(d \cup c_1)^\alpha = d \cup c_2$. Let $w \in S$, then, since $c_1^\alpha = c_2$, we have $(c_1 w)^\alpha = c_1^\alpha w^\alpha = c_2 w^\alpha \in c_2 S$. Since, the choice of $w \in S$ is arbitrary, we have $(c_1 S)^\alpha = c_2 S$. □

Appendix B. Two 2-level regular fractional factorial designs for which Ma et al. (2001)'s sufficiency conjecture fails

Consider the two 2^{10-5} designs d_1 and d_2 given by defining contrast subgroups generated by g_1 and g_2, respectively:

$$g_1 = \{ABF, ACG, ADH, BEI, BCDJ\}$$

$$g_2 = \{ABF, ACG, BDH, CDI, BCEJ\}$$

Designs d_1 and d_2 were found to be non-isomorphic by our graph based isomorphism check, Lin and Sitter (2008)'s eigenvalue check and Clark and Dean (2001)'s isomorphism check for defining contrast subgroups. Table 7 gives the identical CD_2^2 distribution of the two designs.

References

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D., 1999. LAPACK Users' Guide, third ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Bailey, R.A., 1977. Patterns of confounding in factorial designs. Biometrika 64 (3), 597–603.

Bingham, D., Sitter, R.R., 1999. Minimum-aberration two-level fractional factorial split-plot designs. Technometrics 41 (1), 62–70.

Box, G.E.P., Hunter, J.S., 1961. The 2^{k-p} fractional factorial designs. Part I. Technometrics 3 (3), 311–351.

Box, G.E.P., Hunter, J.S., 2000. The 2^{k-p} fractional factorial designs. Part I. Technometrics 42 (1), 28–47.

Cameron, P., Mary, Q., 2004. Automorphisms of graphs. In: Beineke, L.W., Wilson, R.J., Cameron, P.J. (Eds.), Topics in Algebraic Graph Theory. Encyclopedia of Mathematics and its Applications, vol. 102. Cambridge University Press, Cambridge, pp. 137–155.

Chen, J., 1992. Some results on 2^{n-k} fractional factorial designs and search for minimum aberration designs. The Annals of Statistics 20 (4), 2124–2141.

Chen, J., Sun, D.X., Wu, C.F.J., 1993. A catalogue of two-level and three-level fractional factorial designs with small runs. International Statistical Review 61 (1), 131–145.

Cheng, S.-W., Ye, K.Q., 2004. Geometric isomorphism and minimum aberration for factorial designs with quantitative factors. The Annals of Statistics 32 (5), 2168–2185.

Clark, J.B., Dean, A.M., 2001. Equivalence of fractional factorial designs. Statistica Sinica 11, 537–547.

Draper, N.R., Mitchell, T.J., 1967. The construction of saturated 2_r^{k-p} designs. The Annals of Mathematical Statistics 38 (4), 1110–1126.
Draper, N.R., Mitchell, T.J., 1970. Construction of a set of 512-run designs of resolution 5 and a set of even 1024-run designs of resolution 6. The Annals of Mathematical Statistics 41 (3), 876–887.
Fortin, S., 1996. The graph isomorphism problem. Technical Report TR 96-20, Department of Computer Science, The University of Alberta, Edmonton, Alberta, Canada.
Katsaounis, T., Dean, A., 2008. A survey and evaluation of methods for determination of combinatorial equivalence of factorial designs. Journal of Statistical Planning and Inference 138 (1), 245–258.
Kocay, W., 1996. On writing isomorphism programs. In: Wallis, W.D. (Ed.), Computational and Constructive Design Theory, vol. 368. Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 135–175.
Lawson, C., Hanson, R., Kincaid, D., Krogh, F., 1979. Basic linear algebra subprograms for Fortran usage. ACM Transactions on Mathematical Software (TOMS) 5 (3), 308–323.
Lin, C.D., Sitter, R.R., 2008. An isomorphism check for two-level fractional factorial designs. Journal of Statistical Planning and Inference 138 (4), 1085–1101.
Ma, C.X., Fang, K.T., Lin, D.K.J., 2001. On the isomorphism of fractional factorial designs. Journal of Complexity 17 (1), 86–97.
McKay, B.D., 1981. Practical graph isomorphism. Congressus Numerantium 30 (1), 45–87.
McKay, B.D., 1998. Isomorph-free exhaustive generation. Journal of Algorithms 26 (2), 306–324.
McKay, B.D., 2004. nauty User's Guide (Version 2.2). Computer Science Department, Australian National University, ACT 0200, Australia. URL: ⟨http://cs.anu.edu.au/people/Brendan.McKay/nauty/nug.pdf⟩.
Read, R.C., Corneil, D.G., 1977. The graph isomorphism disease. Journal of Graph Theory 1 (1), 339–363.
Stimming, C., 2007. Lapack + + 2.5.2. URL: ⟨http://lapackpp.sourceforge.net/⟩.
Sun, D.X., Li, W., Ye, K.Q., 2002. An algorithm for sequentially constructing non-isomorphic orthogonal designs and its applications. Technical Report SUNYSB-AMS-02-13, Department of Applied Mathematics and Statistics, Stony Brook University, New York.
Wu, C.F.J., Hamada, M., 2000. Experiments: Planning, Analysis, and Parameter Design Optimization. Wiley Series in Probability and Statistics. Wiley, New York.
Xu, H., 2005. A catalogue of three-level regular fractional factorial designs. Metrika 62 (2), 259–281.
Zhu, Y., Zeng, P., 2005. On the coset pattern matrices and minimum m-aberration of 2^{n-p} designs. Statistica Sinica 15 (3), 717.